COMPLEXITY-OPTIMAL AND PARAMETER-FREE FIRST-ORDER METHODS FOR FINDING STATIONARY POINTS OF COMPOSITE OPTIMIZATION PROBLEMS*

WEIWEI KONG†

Abstract. This paper develops and analyzes an accelerated proximal descent method for finding stationary points of nonconvex composite optimization problems. The objective function is of the form $f + h$ where h is a proper closed convex function, f is a differentiable function on the domain of h, and ∇f is Lipschitz continuous on the domain of h. The main advantage of this method is that it is "parameter-free" in the sense that it does not require knowledge of the Lipschitz constant of ∇f or of any global topological properties of f. It is shown that the proposed method can obtain an ε-approximate stationary point with iteration complexity bounds that are optimal, up to logarithmic terms over ε, in both the convex and nonconvex settings. Some discussion is also given about how the proposed method can be leveraged in other existing optimization frameworks, such as min-max smoothing and penalty frameworks for constrained programming, to create more specialized parameter-free methods. Finally, numerical experiments are presented to support the practical viability of the method.

Key words. nonconvex composite optimization, first-order accelerated gradient method, iteration complexity, inexact proximal point method, parameter-free, adaptive, optimal complexity

AMS subject classifications. 47J22, 65K10, 90C25, 90C26, 90C30, 90C60

1. Introduction.

Consider the nonsmooth composite optimization problem

$$\phi_* = \min_{z\in\mathbb{R}^n} \{\phi(z) := f(z) + h(z)\} \tag{1.1}$$

where $h : \mathbb{R}^n \mapsto (\infty, \infty]$ is a proper closed convex function, f is a (possibly nonconvex) continuously differentiable function on an open set containing the domain of h (denoted as $\operatorname{dom} h$), and ∇f is Lipschitz continuous. It is well known that the above assumption on f implies the existence of positive scalars m and M such that

$$-\frac{m}{2}\|x - x'\|^2 \le f(x) - f(x') - \langle\nabla f(x'), x - x'\rangle \le \frac{M}{2}\|x - x'\|^2 \tag{1.2}$$

for every $x, x' \in \operatorname{dom} h$. The quantity (m, M) is often called a *curvature pair* of ϕ (see, for example, [24, 25]), and the first inequality of (1.2) is often called *weak-convexity* when $m > 0$ (see, for example, [8, 9]).

Recently, there has been a surge of interest in developing efficient algorithms for finding ε-stationary points of (1.1), which consist of a pair $(\bar z, \bar v) \in \operatorname{dom} h \times \mathbb{R}^n$ satisfying

$$\bar v \in \nabla f(\bar z) + \partial h(\bar z), \quad \|\bar v\| \le \varepsilon. \tag{1.3}$$

While complexity-optimal algorithms exist for the case where both m and M are known, a *parameter-free* algorithm — one without knowledge of (m, M) — with optimal iteration complexity remains elusive.

*__Funding__: The work of this author has been supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

Versions: v1.0 (May 25, 2022), v2.0 (February 11, 2023), v3.0 (September 17, 2023)

†Work done at the Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN, 37830. **Email:** wwkong92@gmail.com

Our goal in this paper is to develop, analyze, and extend a parameter-free *accelerated proximal descent* (PF.APD) algorithm that obtains, up-to-logarithmic terms, optimal iteration complexities regardless of the convexity of f. Roughly speaking, PF.APD generates a sequence of iterates $\{(z_k, m_k)\} \subseteq \mathrm{dom}\, h \times \mathbb{R}_{++}$ which satisfies

$$(1.4) \qquad z_{k+1} \approx \underset{z \in \mathrm{dom}\, h}{\mathrm{argmin}} \left\{ \frac{\phi(z)}{2m_{k+1}} + \frac{1}{2}\|z - z_k\|^2 \right\}, \quad \phi(z_{k+1}) \le \phi(z_k).$$

for every $k \ge 0$. Notice that the first expression in (1.4) is an inexact *proximal* point update with stepsize $1/(2m_{k+1})$, while the inequality in (1.4) implies $\{\phi(z_k)\}$ is a *descent* sequence. More precisely, the $(k+1)$-th iteration of PF.APD is as follows:

Iteration $k+1$:

(i) Given $\hat{m} \in \mathbb{R}_{++}$, find a *proximal descent* point $z_{k+1} \in \mathrm{dom}\, h$ in which there exists $\hat{u} \in \mathbb{R}^n$ satisfying

$$(1.5) \qquad \hat{u} \in \nabla f(z_{k+1}) + \partial\left(h + \hat{m}\|\cdot - z_k\|^2\right)(z_{k+1}),$$

$$(1.6) \qquad \|\hat{u} + \hat{m}(z_k - z_{k+1})\|^2 \le 2\theta\hat{m}\left[\phi(z_k) - \phi(z_{k+1})\right],$$

$$(1.7) \qquad \|\hat{u}\|^2 \le \hat{m}^2\|z_{k+1} - z_k\|^2,$$

for some $\theta > 0$.

(ii) If a key inequality fails during the execution of step (i), change $\hat{m}$ and try step (i) again. Else, set $m_{k+1} = \hat{m}$.

To find z_{k+1} in step (i) in the above outline, PF.APD specifically applies a parameter-free *accelerated* composite gradient (PF.ACG) algorithm to the subproblem $\min_{z \in \mathrm{dom}\, h}\{\phi(z)/(2\hat{m}) + \|z - z_k\|^2/2\}$ until a finite set of key descent inequalities holds. During the execution of PF.ACG, several inequalities are also checked to ensure its convergence (specifically the ones in (3.5)), and execution is halted if at least one of these inequalities does not hold. These inequalities are always guaranteed to hold when $\hat{m} \ge m$ but may fail to hold when $\hat{m} < m$.

It is worth mentioning that the main difficulties preventing the extension of existing complexity-optimal methods to parameter-free ones is their dependence on *global* topological conditions that strongly depend on the knowledge of (m, M), e.g., (1.2), convexity of f, or knowledge of the Lipschitz modulus of ∇f. Hence, one of the novelties of PF.APD is its ability to relax these conditions to a finite set of *local* topological conditions that only depend on the generated sequence of iterates.

1.1. Literature Review. To keep our notation concise, we will make use of

$$(1.8) \qquad \Delta_0 := \phi(z_0) - \inf_{z \in \mathbb{R}^n} \phi(z), \quad d_0 := \inf_{z_* \in \mathbb{R}^n} \left\{ \|z_0 - z_*\| : \phi(z_*) = \inf_{z \in \mathbb{R}^n} \phi(z) \right\},$$

with the assumption that $\Delta_0 < \infty$ but d_0 may be infinite. Furthermore, we break our discussion between the convex and nonconvex settings and between two types of methods:

I. Algorithms that find $\hat{z} \in \mathrm{dom}\, h$ satisfying $\phi(\hat{z}) - \inf_{z \in \mathbb{R}^n} \phi(z) \le \varepsilon$;

II. Algorithms that find $\bar{z} \in \mathrm{dom}\, h$ satisfying $\mathrm{dist}(0, \nabla f(\bar{z}) + \partial h(\bar{z})) \le \varepsilon$.

It is worth mentioning that complexity-optimal *type-I* methods are not necessarily complexity-optimal *type-II* methods, as noted in [34].

Convex Setting. For this discussion, we assume ϕ to be convex. Paper [32] presents the first complexity-optimal *type-I* methods, under the assumption that $\max\{m, M\}$ is known. Papers [14,15,35,38] (resp. paper [39]) present parameter-free complexity-optimal *type-I* methods for the case of $h \equiv 0$ (resp. h being the indicator of a closed convex set). Paper [1] extends the method in [39] to another parameter-free complexity-optimal *type-I* method for general convex functions h.

The regularized accelerated method described in [34] is one of the earliest nearly-optimal (up to logarithmic terms) *type-II* methods for the case of $h \equiv 0$. However, its complexity is obtained under the strong assumption that: (i) $\max\{m, M\}$ is known, (ii) that there exists $z_* \in \operatorname{dom} h$ such that $\phi(z_*) = \inf_{z\in\mathbb{R}^n} \phi(z)$, (iii) and that a lower bound for d_0 is known. Whether a parameter-free complexity-optimal *type-II* method exists in the convex setting is still unknown.

Nonconvex Setting. For this discussion, we assume ϕ to be nonconvex. One of the most well-known parameter-free *type-II* algorithms is the proximal gradient descent (PGD) method with backtracking line search. In [35], it was shown that this method has a $\mathcal{O}(\varepsilon^{-2})$ *type-II* complexity bound when f is weakly-convex and a suboptimal $\mathcal{O}(\varepsilon^{-1})$ *type-II* bound when f is convex.

One of the earliest accelerated *type-II* methods is found in [12] under the assumption that $\operatorname{dom} h$ is bounded. Following this, paper [13] presented a parameter-free extension of the method in [12] that handles Hölder continuous gradients of f. In a separate line of research, [25] presented a *type-II* accelerated method whose main steps are variants of the (accelerated) FISTA algorithm in [5] and assumes $\operatorname{dom} h$ is bounded. A variant of this method, with improved iteration complexity bounds in the convex setting, was examined in [43]. It is worth noting that some of the methods in [12, 13, 25, 43] have optimal *type-I* bounds when f is convex but all the methods have suboptimal *type-II* bounds even when f is convex.

Motivated by the developments in [12], other papers, e.g., [6,10,23,40], developed similar accelerated methods under different assumptions on f and h. Recently, [18] proposed a parameter-dependent accelerated inexact proximal point (AIPP) method that has an optimal iteration complexity bound of $\mathcal{O}(\sqrt{Mm}\Delta_0/\varepsilon^2)$ when f is weakly convex but has no advantage when f is convex. The work in [19] proposed an adaptive version of AIPP where (m, M) were estimated locally, but a lower bound for $\max\{m, M\}$ was still required. A version of [18] in which the outer proximal point scheme is replaced with an accelerated one was examined in [24], in which a moderately worse iteration-complexity bound was established.

Tangentially Related Works. The developments in [17, 18, 21] strongly influenced and motivated the technical developments of both PF.ACG and PF.APD. Since PF.APD shares strong similarities with AIPP in [18], we mention one of the former's technical improvements on the latter. To begin, note that AIPP is a double-loop method that repeatedly calls an ACG-type method on a sequence of prox subproblems to generate a sequence of *outer* iterates $\{(z_k, v_k, \varepsilon_k)\}$ (at the end of each ACG call) satisfying

$$v_k \in \partial_{\varepsilon_k}\left(\frac{\phi}{2m} + \frac{1}{2}\|\cdot - z_{k-1}\|^2\right)(z_k), \quad \|v_k\|^2 + 2\varepsilon_k \le \sigma^2\|v_k + z_{k-1} - z_k\|^2, \tag{1.9}$$

where $\sigma \in (0,1)$ and $\partial_\varepsilon \psi(x) := \{u \in \mathbb{R}^n : \psi(z') \ge \psi(z) + \langle u, z' - z\rangle - \varepsilon, \quad \forall z' \in \mathbb{R}^n\}$. An expensive refinement procedure, whose effectiveness strongly depends on (1.9) and knowledge of $\max\{m, M\}$, is then applied to each $(z_k, v_k, \varepsilon_k)$ to obtain $(\bar{z}, \bar{v})$

satisfying the inclusion in (1.3). In contrast, the iterates generated at every *inner* iteration of PF.APD always satisfy the inclusion in (1.3), for a different choice of $\bar{v}$ (see Lemma 3.3), and, consequently, the termination of PF.APD can be checked at *every* one of its inner iterations *without* the need for an expensive refinement procedure. It is worth mentioning those relative prox-stationarity criteria, such as (1.7) and (1.9), were previously analyzed in [42] and, more recently, in [2, 26, 28–31].

We now make a brief comparison between PF.APD and two adaptive proximal methods in the literature. First, compared to the redistributed prox-bundle (RPB) method in [16], both PF.APD and RPB are double-loop methods consisting of (i) outer (or "serious") iterations that consider prox-subproblems of the form in (1.4) and some $\lambda > 0$ and (ii) inner (or "null") iterations that consider composite subproblems of the form $\min_{y\in\mathbb{R}^n}\{\Phi_{j,k}(y) + h(y)\}$ for the k-th subproblem and j-th iteration, until there is a sufficient decrease in $\phi(z_k)$. However, PF.APD chooses $\Phi_{j,k}$ to be a quadratic approximation of Φ_k centered on a specially chosen point (see the update of y_{k+1} in Algorithm 3.1), while RPB chooses $\Phi_{j,k}$ to be the maximum of a different set of quadratic approximations, which is generally more difficult to minimize. Moreover, PF.APD uses values of $\nabla f(\cdot)$ and elements of $\partial h(\cdot)$ in its construction of $\Phi_{j,k}$ whereas RPB uses elements of the limiting subdifferential of ϕ.

Second, compared to the Catalyst Acceleration Framework (CAF) in [40], both PF.APD and CAF consider inexactly solving proximal subproblems as in (1.4) using ACG subroutine and subproblem termination conditions similar to (2.3)–(2.4). However, CAF obtains the inequality in (1.4) by inexactly solving a second prox-subproblem (with a different prox center) and applying an extra interpolation step. As a consequence, CAF requires nearly double the work of PF.APD. Moreover, the line search strategy (analogous to Algorithm 3.1 and Algorithm 3.3) employed by CAF in [40, Algorithm 3] is static in that it prescribes a large number of ACG iterations, whereas the line search strategy in PF.APD is dynamic in that it checks a finite set of simple inequalities at each ACG iteration.

1.2. Contributions. Throughout, we refer to the two types of algorithms described in the previous subsection. Given a starting point $z_0 \in \operatorname{dom} h$ and a tolerance $\varepsilon > 0$, it is shown that PF.APD has the following nice properties:

(i) for any $\hat{m} > 0$, it always obtains a pair $(\bar{z}, \bar{v}) \in \operatorname{dom} h \times \mathbb{R}^n$ satisfying (1.3);
(ii) if f is nonconvex, then it stops in $\tilde{\mathcal{O}}(\sqrt{mM\Delta_0}/\varepsilon^2)$ resolvent evaluations[1];
(iii) if f is convex, then it stops in $\tilde{\mathcal{O}}(\sqrt{M}\min\{\sqrt{\Delta_0}/\varepsilon, d_0/\sqrt{\varepsilon}\})$ resolvent evaluations;

Both of the above complexity bounds are optimal (up to logarithmic terms in) in terms of Δ_0, M, m, and ε (although suboptimal by a factor of $\sqrt{d_0}$ in the convex case). Moreover, it appears to be the first time that a *type-II* parameter-free method has obtained such bounds[2]. Improved iteration complexity bounds are also obtained when d_0 is known. Also, all of the above results are obtained under the mild assumption that the optimal value in (1.1) is finite and does not assume the boundedness of $\operatorname{dom} h$ (cf. [25, 43]) nor that an optimal solution of (1.1) exists.

For convenience, we compare in Table 1.1 the best iteration complexity bounds of some of the parameter-free methods listed in the previous subsection with two instances of PF.APD. For shorthands, PGD is the adaptive proximal gradient descent method in [35], UPF is the UPFAG method in [13], ANCF is the ADAP-NC-FISTA method in [25], VRF is the VAR-FISTA method in [43], and APD is as in Algo-

[1] The notation $\tilde{\mathcal{O}}(\cdot)$ ignores any terms that logarithmically depend on the tolerance ε.

[2] Compare this to the complexity-optimal methods in [34] and [18] which require knowledge of d_0 and (m, M), respectively.

rithm 3.4 in this paper with $m_0 = 1$.

Algorithm	f **convex**	f **nonconvex**	$D_h < \infty$
PGD [35]	$\mathcal{O}\left(\frac{M^{3/2} d_0}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{M^2 \Delta_0}{\varepsilon^2}\right)$	No
UPF [13]	N/A	$\mathcal{O}\left(\frac{M\Delta_0}{\varepsilon^2}\right)$	No
ANCF [25]	$\mathcal{O}\left(\frac{M^{2/3}[\Delta_0^{1/3} + d_0^{2/3}]}{\varepsilon^{2/3}} + \frac{MD_h}{\varepsilon}\right)$	$\mathcal{O}\left(mM^2\left[\frac{mD_h^2 + \Delta_0}{\varepsilon^2}\right]\right)$	Yes
VRF [43]	$\mathcal{O}\left(\frac{M^{2/3}[\Delta_0^{1/3} + D_h^{2/3}]}{\varepsilon^{2/3}}\right)$	$\mathcal{O}\left(mM^2 D_h^2\left[\frac{1+m^2}{\varepsilon^2}\right]\right)$	Yes
APD	$\tilde{\mathcal{O}}\left(\sqrt{M}\left[\min\left\{\frac{\sqrt{\Delta_0}}{\varepsilon}, \frac{d_0}{\sqrt{\varepsilon}}\right\}\right]\right)$	$\tilde{\mathcal{O}}\left(\frac{\sqrt{mM}\Delta_0}{\varepsilon^2}\right)$	No
Known Lower Bounds	$\Omega\left(\sqrt{M}\left[\min\left\{\frac{\sqrt{\Delta_0}}{\varepsilon}, \sqrt{\frac{d_0}{\varepsilon}}\right\}\right]\right)$	$\Omega\left(\frac{\sqrt{mM}\Delta_0}{\varepsilon^2}\right)$	-

TABLE 1.1

Lower bounds and iteration-complexity bounds of various parameter-free type-II *composite optimization algorithms for finding ε-stationary points as in (1.3). The scalar D_h denotes the diameter of* $\operatorname{dom} h$ *and it is assumed that d_0, Δ_0, m, and M are not known but M is greater than or equal to m for the listed algorithms. The lower bounds for the convex (resp. nonconvex) case can be found in [7, Theorem 1] (resp. [48, Theorem 4.5]).*

Notice that the analysis for UPFAG does not include an iteration complexity bound for finding stationary points when f is convex, while ANCF and VRF suffer from the requirement that $\operatorname{dom} h$ must be bounded. Moreover, up until this point, PGD was the only parameter-free *type-II* algorithm with an established iteration complexity bound for the unbounded case when f is convex. None of the parameter-free methods before this work, in the nonconvex setting, could obtain the optimal complexity bound in [18].

In addition to the development of PF.APD, some details are given regarding how PF.APD could be used in other existing optimization frameworks, including min-max smoothing and penalty frameworks for constrained optimization. The main advantages of these resulting frameworks are that (i) they are parameter-free and (ii) they have improved complexities when f in (1.1) is convex, without requiring any adjustments to their inputs.

Finally, numerical experiments are given to support the practical efficiency of PF.ADP on some randomly generated problem instances. These experiments specifically show that PF.APD consistently outperforms several existing parameter-free methods in practice.

1.3. Organization. Section 2 presents background material. Section 3 presents PF.ACG, PF.APD, and their iteration complexity bounds. Section 4 gives the proofs of several important technical results. Section 5 describes how PF.APD can be used in existing optimization frameworks. Section 6 presents some numerical experiments. Section 7 gives some concluding remarks. Several technical appendices follow after the above sections.

1.4. Notation and Basic Definitions. $\mathbb{R}_+$ and $\mathbb{R}_{++}$ denote the set of nonnegative and positive real numbers, respectively. $\mathbb{R}^n$ denotes an n-dimensional Euclidean space with inner product and norm denoted by $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$, respectively. $\operatorname{dist}(x, X)$ denotes the Euclidean distance of a point x to a set X. For any $t > 0$, we denote $\log^{+1}(t) := \max\{\log t, 1\}$. For a function $h : \mathbb{R}^n \to (-\infty, \infty]$ we denote $\mathrm{dom}\, h := \{x \in \mathbb{R}^n : h(x) < +\infty\}$ to be the domain of h. Moreover, h is considered proper if $\mathrm{dom}\, h \neq \emptyset$. The set of all lower semi-continuous proper convex functions defined in $\mathbb{R}^n$ is denoted by $\overline{\mathrm{Conv}}\,(\mathbb{R}^n)$. The convex subdifferential of a proper function $h : \mathbb{R}^n \to (-\infty, \infty]$ is given by

$$\partial h(z) := \{u \in \mathbb{R}^n : h(z') \geq h(z) + \langle u, z' - z\rangle, \quad \forall z' \in \mathbb{R}^n\} \tag{1.10}$$

for every $z \in \mathbb{R}^n$. If ψ is a real-valued function which is differentiable at $\bar{z} \in \mathbb{R}^n$, then its affine/linear approximation $\ell_\psi(\cdot, \bar{z})$ at $\bar{z}$ is given by

$$\ell_\psi(z; \bar{z}) := \psi(\bar{z}) + \langle \nabla\psi(\bar{z}), z - \bar{z}\rangle \quad \forall z \in \mathbb{R}^n. \tag{1.11}$$

2. Background.

This section gives some necessary background for presenting PF.ACG and PF.APD. More specifically, Subsection 2.1 describes and comments on the problem of interest, while Subsection 2.2 presents a general proximal descent scheme which serves as a template for PF.APD.

2.1. Problem of Interest.

To reiterate, we are interested in the following composite optimization problem:

Problem $\mathcal{CO}$: Given $\varepsilon \in \mathbb{R}_{++}$ and a function $\phi = f + h$ satisfying:

- ⟨A1⟩ $h \in \overline{\mathrm{Conv}}\,(\mathbb{R}^n)$ and the resolvent $(\lambda\partial h + \mathrm{id})^{-1}$ is easy to compute for any $\lambda > 0$,
- ⟨A2⟩ f is continuously differentiable on an open set $\Omega \supseteq \mathrm{dom}\, h$, and ∇f is $\mathcal{M}$-Lipschitz continuous on $\mathrm{dom}\, h$ for some $\mathcal{M} \in \mathbb{R}_{++}$,
- ⟨A3⟩ $\phi_* = \inf_{z \in \mathbb{R}^n} \phi(z) > -\infty$,

find a pair $(\bar{z}, \bar{v}) \in \mathrm{dom}\, h \times \mathbb{R}^n$ satisfying (1.3).

Of the three above assumptions, only ⟨A1⟩ is a necessary condition that is used to ensure PF.APD is well-defined. Assumptions ⟨A2⟩–⟨A3⟩, on the other hand, are sufficient conditions that are used to show that PF.APD stops in a finite number of iterations. It is possible to replace assumption ⟨A2⟩ with more general smoothness conditions (e.g., Hölder continuity [13,36]) at the cost of a possibly more complicated analysis. It is known[3] that assumption ⟨A2⟩ holds if and only if

$$|f(z) - \ell_f(z; z')| \leq \frac{\mathcal{M}}{2}\|z - z'\|^2, \quad \forall z, z' \in \mathrm{dom}\, h, \tag{2.1}$$

which implies $(\mathcal{M}, \mathcal{M})$ is a curvature pair of ϕ.

We now comment on criterion (1.3). First, it is related to the directional derivative of ϕ:

$$\begin{aligned}\min_{\|d\|=1} \phi'(z; d) &= \min_{\|d\|=1} \max_{\zeta \in \partial h(z)} \langle \nabla f(z) + \zeta, d\rangle = \max_{\zeta \in \partial h(z)} \min_{\|d\|=1} \langle \nabla f(z) + \zeta, d\rangle \\ &= -\min_{\zeta \in \partial h(z)} \|\nabla f(z) + \zeta\| = -\mathrm{dist}(0, \nabla f(z) + \partial h(z)).\end{aligned}$$

[3]The proof of the forward direction is well-known (see, for example, [4,37]) while the proof of the reverse direction can be found, for example, in [17, Proposition 2.1.55]. For the special case where f is convex and real-valued, the proof of the reverse direction can be found, for example, in [3, Theorem 18.15] and [33, 2.1.5].

Consequently, if $\bar{z} \in \operatorname{dom} h$ is a local minimum of ϕ then $\min_{\|d\|=1} \phi'(\bar{z}; d) \geq 0$ and the above relation implies that (1.3) holds with $\varepsilon = 0$. That is, (1.3) is a necessary condition for local optimality of a point $\bar{z} \in \operatorname{dom} h$. Second, when f is convex then (1.3) with $\varepsilon = 0$ implies that $0 \in \nabla f(\bar{z}) + \partial h(\bar{z}) = \partial \phi(\bar{z})$ and $\bar{z}$ is a global minimum. Given the first comment, (1.3) is equivalent to global optimality of a point $\bar{z} \in \operatorname{dom} h$ when f is convex.

2.2. General Proximal Descent Scheme. Our interest in this subsection is the general proximal descent scheme in Algorithm 2.1, which follows the ideas in (1.5)–(1.7). Its iteration scheme serves as a template for the PF.APD presented in Subsection 3.2.

Algorithm 2.1 General Proximal Descent Scheme

`Data:` (f, h) as in ⟨`A1`⟩–⟨`A3`⟩, $z_0 \in \operatorname{dom} h$;
`Parameters:` $\theta \in \mathbb{R}_+$;
1: **for** $k \leftarrow 0, 1, \ldots$ **do**
2: **find** $(z_{k+1}, u_{k+1}) \in \operatorname{dom} h \times \mathbb{R}^n$ and $m_{k+1} \in \mathbb{R}_{++}$ satisfying

$$u_{k+1} \in \nabla f(z_{k+1}) + 2m_{k+1}(z_{k+1} - z_k) + \partial h(z_{k+1}), \tag{2.2}$$

$$\|u_{k+1} + 2m_{k+1}(z_k - z_{k+1})\|^2 \leq 2\theta m_{k+1} \left[\phi(z_k) - \phi(z_{k+1})\right], \tag{2.3}$$

$$\|u_{k+1}\|^2 \leq m_{k+1}^2 \|z_{k+1} - z_k\|^2. \tag{2.4}$$

Before presenting the properties of Algorithm 2.1, let us comment on its steps. First, (2.2)–(2.4) are analogous to (1.5)–(1.7) because of assumption ⟨`A1`⟩. Second, if $f + m_{k+1}\|\cdot\|^2$ is convex and $u_{k+1} = 0$ then (2.2) implies that

$$z_{k+1} = \operatorname*{argmin}_{z \in \operatorname{dom} h} \left\{ \frac{\phi(z)}{2m_{k+1}} + \frac{1}{2}\|z - z_{k+1}\|^2 \right\},$$

which is a proximal point update with stepsize $1/(2m_{k+1})$. Third, (2.3) implies that Algorithm 2.1 is a descent scheme, i.e., $\phi(z_{k+1}) \leq \phi(z_k)$ for $k \geq 0$. Hence, in view of the second comment, this justifies its qualifier as a "proximal descent" scheme.

It is also worth mentioning that (2.3)–(2.4) are similar to conditions in existing literature. More specifically, a version of (2.3) can be found in the descent scheme of [19], while an inequality similar to (2.4) can be found in the GIPP framework of [18] with $\sigma = 1$, $\tilde{\varepsilon} = 0$, and $v_{k+1} = u_{k+1}/m_{k+1}$. However, the addition of condition (2.2) appears to be new.

We now present the most important properties of Algorithm 2.1. The first result supports the importance of conditions (2.2)–(2.3).

LEMMA 2.1. *Given $z_0 \in X$, let $\{(z_{k+1}, u_{k+1})\}_{k \geq 0}$ denote a sequence of iterates satisfying (2.2)–(2.3). Moreover, let Δ_0 be as in (1.8), and define*

$$v_{k+1} := u_{k+1} + 2m_{k+1}(z_k - z_{k+1}), \quad \Lambda_{k+1} := \sum_{j=0}^{k} \frac{1}{m_{j+1}}, \quad \forall k \geq 0.$$

Then, for every $k \geq 0$,

(a) $v_{k+1} \in \nabla f(z_{k+1}) + \partial h(z_{k+1})$;
(b) $\min_{0 \leq j \leq k} \|v_{j+1}\|^2 \leq 2\theta \Delta_0 \Lambda_{k+1}^{-1}$.

Proof. (a) This follows immediately from (2.2) and the definition of v_{k+1}.

(b) Summing up both sides of (2.3) from 0 to k, the definition of v_{k+1}, and the definition of ϕ_*, we have that

$$\Lambda_{k+1} \min_{0 \le j \le k} \|v_{j+1}\|^2 \le \sum_{j=0}^{k} \frac{\|v_{j+1}\|^2}{m_{j+1}} \overset{(2.3)}{\le} 2\theta \sum_{j=0}^{k} [\phi(z_j) - \phi(z_{j+1})]$$
$$= 2\theta [\phi(z_0) - \phi(z_{k+1})] \le 2\theta [\phi(z_0) - \phi_*] = 2\theta \Delta_0. \qquad \square$$

Notice that Lemma 2.1(b) implies that if $\lim_{k\to\infty} \Lambda_{k+1} \to \infty$ then we have that $\lim_{k\to\infty} \min_{j\le k} \|v_{j+1}\| \to 0$. Moreover, if $\sup_{k\ge 0} m_{k+1} < \infty$ then for any $\varepsilon > 0$, there exists some finite $j \ge 0$ such that $\|v_{j+1}\| \le \varepsilon$.

The next result shows that if m_{k+1} is bounded relative to the global topology of f, and conditions (2.2)–(2.4) hold, then a more refined bound of $\min_{j\le k} \|v_{j+1}\|$ can be obtained. To keep the notation concise, we make use of the following quantity:

$$R_\tau(\hat{z}) := \inf_{z\in\mathbb{R}^n} \left\{ R_\tau(z, \hat{z}) := \frac{\phi(z) - \phi_*}{\tau} + \frac{1}{2}\|z - \hat{z}\|^2 \right\}. \tag{2.5}$$

It is easy to see that $R_\tau(z')$ is the Moreau envelope of ϕ/τ at z' shifted by $-\phi_*/\tau$.

LEMMA 2.2. *Given $z_0 \in X$, let $\{(v_{j+1}, z_{j+1}, \Lambda_{j+1})\}_{j\ge 0}$ be as in Lemma 2.1 and $k \ge 0$ be fixed. Moreover, suppose (2.4) holds and that there exists $\tilde{m} > 0$ such that $f + \tilde{m}\|\cdot\|^2/2$ is convex. If $\min_{0\le j\le k} m_{j+1} \ge \tilde{m}$ and $\max_{0\le j\le k} m_{j+1} \le (1+\nu)\tilde{m}$ for some $\nu > 0$, then*

$$\phi(z_{k+1}) + \frac{m_{k+1}}{2}\|z_{k+1} - z_k\|^2 \le \inf_{z\in\mathbb{R}^n} \left\{ \phi(z) + \frac{\nu\tilde{m}}{2}\|z - z_k\|^2 \right\}, \tag{2.6}$$

and if $k \ge 1$ then it holds that

$$\min_{1\le j\le k} \|v_{j+1}\|^2 \le 2\theta\nu\tilde{m} \left[\frac{R_{\nu\tilde{m}}(z_0)}{\Lambda_{k+1} - m_1^{-1}} \right]. \tag{2.7}$$

Proof. Using the assumption that $m_{k+1} \ge \tilde{m}$ and (2.2), we have that $f(\cdot) + m_{k+1}\|\cdot - z_k\|^2$ is $\tilde{m}$-strongly convex and, hence,

$$u_{k+1} \in \nabla f(z_{k+1}) + 2m_{k+1}(z_{k+1} - z_k) + \partial h(z_{k+1})$$
$$= \nabla f(z_{k+1}) - \tilde{m}(z_{k+1} - z_{k+1}) + 2m_{k+1}(z_{k+1} - z_k) + \partial h(z_{k+1})$$
$$= \partial \left(\phi - \frac{\tilde{m}}{2}\|\cdot - z_{k+1}\|^2 + m_{k+1}\|\cdot - z_k\|^2 \right)(z_{k+1}). \tag{2.8}$$

Using (2.8), (2.4), and the bound $\langle a, b\rangle \ge -\|a\|^2/(2m_{k+1}) - m_{k+1}\|b\|^2/2$ for any $a, b \in \mathbb{R}^n$, it holds for any $z \in \mathbb{R}^n$ that

$$\phi(z) + m_{k+1}\|z - z_k\|^2$$
$$\overset{(2.8)}{\ge} \phi(z_{k+1}) + m_{k+1}\|z_k - z_{k+1}\|^2 + \frac{\tilde{m}}{2}\|z - z_{k+1}\|^2 + \langle u_{k+1}, z - z_{k+1}\rangle$$
$$\ge \phi(z_{k+1}) + m_{k+1}\|z_k - z_{k+1}\|^2 - \frac{1}{2m_{k+1}}\|u_{k+1}\|^2 + \frac{\tilde{m} - m_{k+1}}{2}\|z - z_{k+1}\|^2$$
$$\overset{(2.4)}{\ge} \phi(z_{k+1}) + \frac{m_{k+1}}{2}\|z_k - z_{k+1}\|^2 + \frac{\tilde{m} - m_{k+1}}{2}\|z - z_{k+1}\|^2.$$

Re-arranging terms and using the assumption $m_{k+1} \le (1+\nu)\tilde{m}$, we then have that

$$\phi(z_{k+1}) + \frac{m_{k+1}}{2}\|z_k - z_{k+1}\|^2 \le \phi(z) + \frac{\nu\tilde{m}}{2}\|z - z_k\|^2,$$

which implies (2.6) as $z \in \mathbb{R}^n$ was arbitrary. To show (2.7), we use (2.6) at $k = 1$, (2.3), and the definition of v_{k+1} to conclude that

$$\begin{aligned} R_{\nu\tilde{m}}(z_0) &= \inf_{z\in\mathbb{R}^n}\left\{\frac{\phi(z)-\phi_*}{\nu\tilde{m}} + \frac{1}{2}\|z - z_0\|^2\right\} \ge \frac{\phi(z_1)-\phi_*}{\nu\tilde{m}} + \frac{m_1}{2\nu\tilde{m}}\|z_1 - z_0\|^2 \\ &\overset{(2.6)}{\ge} \frac{\phi(z_1)-\phi(z_{k+1})}{\nu\tilde{m}} = \frac{\sum_{j=1}^{k}[\phi(z_j)-\phi(z_{j+1})]}{\nu\tilde{m}} \overset{(2.3)}{\ge} \frac{1}{2\theta\nu\tilde{m}}\sum_{j=1}^{k}\frac{\|v_{j+1}\|^2}{m_{j+1}} \\ &\ge \frac{\sum_{j=1}^{k} m_{j+1}^{-1}}{2\theta\nu\tilde{m}}\left(\inf_{1\le j\le k}\|v_{j+1}\|^2\right) = \frac{\Lambda_{k+1} - m_1^{-1}}{2\theta\nu\tilde{m}}\inf_{1\le j\le k}\|v_{j+1}\|^2. \qquad \square \end{aligned}$$

Similar to the previous lemma, the above result also implies that if $\lim_{k\to\infty}\Lambda_{k+1} \to \infty$ then we have $\lim_{k\to\infty}\min_{j\le k}\|v_{j+1}\| \to 0$. However, it is more general in the sense that the rate of convergence depends on $R_{\nu\tilde{m}}(z_0)$ instead of Δ_0, and the former can be bounded as

$$R_{\nu\tilde{m}}(z_0) \le \min\{R_{\nu\tilde{m}}(z_0, z_0), R_{\nu\tilde{m}}(z_*, z_0)\} \le \min\left\{\frac{\Delta_0}{\nu\tilde{m}}, \frac{d_0^2}{2}\right\}, \tag{2.9}$$

where z_* is any optimal solution of (1.1) that is the closest to z_0 and (Δ_0, d_0) are as in (1.8). This fact will be important when we establish an iteration complexity bound for PF.APD in the convex setting.

3. Parameter-Free Algorithms.

This section presents PF.ACG, PF.APD, and their iteration complexity bounds. More specifically, Subsection 3.1 presents PF.ACG, while Subsection 3.2 presents PF.APD.

It is also worth recalling that PF.APD is an implementation of the general descent scheme of the previous section that repeatedly calls PF.ACG to obtain a single iteration of the scheme mentioned above.

3.1. PF.ACG Algorithm.

Broadly speaking, PF.ACG is a modification of the well-known FISTA [5, 11] algorithm for minimizing μ-strongly convex composite functions. Specifically, both PF.ACG and FISTA consider the composite optimization problem

$$\min_{x\in\mathbb{R}^n}\{\psi(x) := \psi^s(x) + \psi^n(x)\} \tag{3.1}$$

where (ψ^s, ψ^n) satisfies the following assumptions:

- ⟨B1⟩ $\psi^n \in \overline{\text{Conv}}\,(\mathbb{R}^n)$ and the resolvent $(\lambda\partial\psi^n + \text{id})^{-1}$ is easy to compute for any $\lambda > 0$,
- ⟨B2⟩ ψ^s is continuously differentiable on an open set $\Omega \supseteq \text{dom}\,\psi^n$, and $\nabla\psi^s$ is L_*-Lipschitz continuous on $\text{dom}\,\psi^n$ for some $L_* \in \mathbb{R}_{++}$.

Similar to (2.1), note that ⟨B2⟩ implies

$$|\psi^s(x) - \ell_{\psi^s}(x; x')| \le \frac{L_*}{2}\|x - x'\|^2 \quad \forall x, x' \in \text{dom}\,\psi^n. \tag{3.2}$$

PF.ACG differs from FISTA in that it adds two stopping conditions that help implement a single iteration of Algorithm 2.1. Specifically, for a given function pair

(f,h) satisfying ⟨A1⟩–⟨A2⟩, hyperparameters $(\sigma,\theta,\mu) \in \mathbb{R}^3_{++}$, and an initial point $\hat{z} \in \mathrm{dom}\ h$, if PF.ACG is invoked with

$$\psi^s(\cdot) = \frac{f(\cdot)}{2\hat{m}} + \frac{1}{2}\|\cdot - \hat{z}\|^2, \quad \psi^n(\cdot) = \frac{h(\cdot)}{2\hat{m}}, \tag{3.3}$$

for some $\hat{m} > 0$, then either (i) PF.ACG has found a pair (y,u) satisfying conditions (2.2)–(2.4) with $(z_{k+1}, u_{k+1}, m_{k+1}, z_k) = (y, u, m, \hat{z})$, or (ii) some local μ-strong convexity condition has failed, and the estimate of μ or the function pair (ψ^s, ψ^n) has to be changed.

We now present the details of PF.ACG and its key properties. To help our discussion, we first give the complete pseudocode of PF.ACG through Algorithm 3.1 and Algorithm 3.2. More specifically, Algorithm 3.1 presents the accelerated gradient FISTA update and (Lipschitz constant) line search strategy used in PF.ACG, while Algorithm 3.2 describes the other steps of PF.ACG and how Algorithm 3.1 is invoked.

Algorithm 3.1 Line Search and Accelerated Gradient Step Subroutine

---

`Data`: (ψ^s, ψ^n) as in ⟨B1⟩–⟨B2⟩, $(\hat{y}, \hat{x}) \in \mathrm{dom}\,\psi^n \times \mathbb{R}^n$, $\hat{A} \geq 0$, $\mu \in \mathbb{R}_{++}$, $\hat{L} \in [\mu, \infty)$;
`Hyper-parameters:` $\beta \in (1, \infty)$;
`Outputs`: $(A, \tilde{x}, y, x, L) \in \mathbb{R}_+ \times \mathbb{R}^n \times \mathrm{dom}\ \psi^n \times \mathbb{R}^n \times \mathbb{R}_+$ and function q;

$\psi \leftarrow \psi^s + \psi^n$
for $\ell \leftarrow 0, 1, \ldots$ **do**
  $L \leftarrow \hat{L}\beta^\ell$
  `▷ Step 1: Accelerated gradient step.`
  $\xi \leftarrow 1 + \mu\hat{A}$ and find $\hat{a}$ satisfying $\hat{a}^2 = \hat{\xi}(\hat{a} + \hat{A})/L$
  $A \leftarrow \hat{A} + \hat{a}$
  $\tilde{x} \leftarrow \frac{\hat{A}}{A}\hat{y} + \frac{\hat{a}}{A}\hat{x}$
  $y \leftarrow \mathrm{argmin}_{r \in \mathbb{R}^n} \left\{ \ell_{\psi^s}(z; \tilde{x}) + \psi^n(z) + \frac{L+\mu}{2}\|z - \tilde{x}\|^2 \right\}$
  $x \leftarrow \hat{x} + \frac{\hat{a}}{1 + A\mu}\left[L(y - \tilde{x}) + \mu(y - \hat{x})\right]$
  `▷ Step 2: Descent condition check.`
  if the inequality

$$\psi^s(y) - \ell_{\psi^s}(y; \tilde{x}) \leq \tfrac{L}{2}\|y - \tilde{x}\|^2 \tag{3.4}$$

  holds, then **return** $(A, \tilde{x}, y, x, L)$

We next present some key properties about Algorithm 3.2 and its iterates. As their proof is mostly technical, we moved it to Subsection 4.1.

LEMMA 3.1. *For every $j \geq 0$,*
(a) $A_{j+1} \geq (1/L_0)\prod_{i=1}^{j}[1 + \sqrt{\mu/(2L_i)}]$ *and*

$$L_j \leq L_{j+1} \leq \bar{L} := \max\{1, \alpha L_*\}. \tag{3.7}$$

(b) $r_{j+1} \in \nabla\psi^s(y_{j+1}) + \partial\psi^n(y_{j+1})$*;*
(c) if ψ^s is μ-strongly convex, then (3.5) *holds;*
(d) if (3.5) *holds and*

$$A_{j+1} \geq \frac{16\bar{L}^2}{\mu}\max\left\{\frac{1}{\sigma^2}, \frac{4\theta}{\theta - 2}\right\} =: \mathcal{A}_{\mu,\bar{L}}(\sigma, \theta) \tag{3.8}$$

then (3.6) *holds.*

Algorithm 3.2 Parameter-Free Accelerated Composite Gradient (PF.ACG) Algorithm

`Data:` (ψ^s, ψ^n) as in ⟨B1⟩–⟨B2⟩, $y_0 \in \operatorname{dom} \psi^n$, $\mu \in \mathbb{R}_{++}$, $L_0 \in [\mu, \infty)$;
`Hyper-parameters:` $\sigma \in \mathbb{R}_{++}$, $\theta \in (2, \infty)$, $\beta \in (1, \infty)$;
`Outputs:` $(y_{j+1}, u_{j+1}, L_{j+1}) \in \operatorname{dom} \psi^n \times \mathbb{R}^n \times \mathbb{R}_{++}$;

1: $(x_0, A_0) \leftarrow (y_0, 0)$
2: $\psi(\cdot) \leftarrow \psi^s(\cdot) + \psi^n(\cdot)$
3: **for** $j \leftarrow 0, 1, \ldots$ **do**
  ▷ `Step 1: Line search for` L_{j+1} `and accelerated gradient step.`
4: **call** Algorithm 3.1 with data (ψ^s, ψ^n), $(\hat{y}, \hat{x}) \equiv (y_j, x_j)$, $\hat{A} \equiv A_j$, $\hat{\xi} \equiv \xi_j$, μ, $\hat{L} \equiv L_j$ and hyper-parameter β to obtain $(A_{j+1}, \tilde{x}_j, y_{j+1}, x_{j+1}, L_{j+1})$
  ▷ `Step 2: "Bad" termination check.`
5: $r_{j+1} \leftarrow \nabla\psi^s(y_{j+1}) - \nabla\psi^s(\tilde{x}_j) + (L_{j+1} + \mu)(\tilde{x}_j - y_{j+1})$
6: **if** the inequalities

$$\begin{aligned} \mu A_{j+1}\|y_{j+1} - \tilde{x}_j\|^2 &\leq \|y_{j+1} - y_0\|^2, \\ \psi(y_0) &\geq \psi(y_{j+1}) + \langle r_{j+1}, y_0 - y_{j+1} \rangle, \end{aligned} \tag{3.5}$$

  do not hold, then **return** $(y_{j+1}, r_{j+1}, L_{j+1})$
  ▷ `Step 3: "Good" termination check.`
7: **if** the inequalities

$$\begin{aligned} \|r_{j+1}\|^2 &\leq \sigma^2\|y_{j+1} - y_0\|^2, \\ \|r_{j+1} + y_0 - y_{j+1}\|^2 &\leq \theta\left[\psi(y_0) - \psi(y_{j+1}) + \tfrac{1}{2}\|y_{j+1} - y_0\|^2\right], \end{aligned} \tag{3.6}$$

  hold, then **return** $(y_{j+1}, r_{j+1}, L_{j+1})$

We now give a complexity bound for Algorithm 3.2 and a condition for guaranteeing its successful termination.

Proposition 3.2. *The following properties hold about Algorithm 3.2:*

(a) it stops in

$$\left\lceil 1 + 2\sqrt{\frac{2\bar{L}}{\mu}} \log^{1+}\left\{\bar{L}\mathcal{A}_{\mu,\bar{L}}(\sigma, \theta)\right\}\right\rceil, \tag{3.9}$$

where $\bar{L}$ and $\mathcal{A}_{\mu,\bar{L}}$ are as in (3.7) and (3.8), respectively.

(b) if ψ^s is μ-strongly convex, then it always terminates in its Step 3 with a triple $(y_{j+1}, u_{j+1}, L_{j+1})$ satisfying (3.6) and $L_0 \leq L_{j+1} \leq \bar{L}$.

Proof. (a) Let $J + 1$ denote the quantity in (3.9) and suppose Algorithm 3.2 has not terminated at the end of iteration $J+1$. Moreover, denote $\mathcal{A} := \mathcal{A}_{\mu,\bar{L}}(\sigma, \theta)$. Using Lemma 3.1(a), we first have

$$A_{J+1} \geq \frac{1}{L_0}\prod_{i=1}^{J}\left(1 + \sqrt{\frac{\mu}{2L_i}}\right) \geq \frac{1}{\bar{L}}\left(1 + \sqrt{\frac{\mu}{2\bar{L}}}\right)^J \tag{3.10}$$

Using the above bound, the fact that $J \geq 2\sqrt{2\bar{L}/\mu}\log(\bar{L}\mathcal{A})$ from the definition in (3.9), the bound $\mu \leq \bar{L}$, and the fact that $\log(1 + t) \geq t/2$ on $t \in [0, 1]$, it holds that

$$\log(\bar{L}\mathcal{A}) \leq \frac{J}{2}\sqrt{\frac{\mu}{2\bar{L}}} \leq J\log\left(1 + \sqrt{\frac{\mu}{2\bar{L}}}\right) \overset{(3.10)}{\leq} \log(\bar{L}A_{J+1})$$

which implies $A_{J+1} \geq \mathcal{A}$. Hence, it follows from Lemma 3.1(d) that (3.6) holds. In view of Step 3 of Algorithm 3.2 this implies that termination has to have occurred at or before iteration $J+1$, which contradicts our initial assumption. Thus, Algorithm 3.2 must have terminated by iteration $J+1$.

(b) This follows immediately from part (a) and Lemma 3.1(c). ∎

The last result of this subsection shows how to invoke Algorithm 3.2 so that its successful termination implements a single iteration of Algorithm 2.1.

LEMMA 3.3. *Suppose Algorithm 3.2 is called with* (ψ^s, ψ^n) *as in* (3.3) *for some* $m > 0$ *and* $\hat{z} \in \operatorname{dom} \psi^n$, $\sigma = 1/4$, *and* $y_0 = \hat{z}$. *If the call terminates in Step 3 with an output triple* $(y_{j+1}, r_{j+1}, L_{j+1})$, *then the quadruple* $(z_{k+1}, u_{k+1}, m_{k+1}, z_k) = (y_{j+1}, 2mr_{j+1}, m, \hat{z})$ *satisfies* (2.2)–(2.4).

Proof. Using Lemma 3.1(b), it holds that

$$\begin{aligned} u_{k+1} = 2mr_{j+1} &\in 2m\left[\nabla\psi^s(y_{j+1}) + \partial\psi^n(y_{j+1})\right] \\ &= \nabla f(z_{k+1}) + 2m_{k+1}(z_{k+1} - z_k) + \partial\psi^n(z_{k+1}). \end{aligned}$$

which is exactly (2.2). Now, using the first inequality in (3.6), the choice of $\sigma = 1/4$, and the fact that $y_0 = \hat{z} = z_k$, we have

$$\|u_{k+1}\|^2 = 4m^2\|r_{j+1}\|^2 \overset{(3.6)}{\leq} m^2\|y_{j+1} - y_0\|^2 = m_{k+1}^2\|z_{k+1} - z_k\|^2,$$

which is exactly (2.4). Finally, the second condition of (3.6), the relation $\psi(\cdot) = \phi(\cdot)/(2m_{k+1}) + \|\cdot - y_0\|^2/2$, and the fact that $y_0 = \hat{z} = z_k$ imply

$$\begin{aligned} &\|u_{k+1} + 2m_{k+1}(z_k - z_{k+1})\|^2 = 4m_{k+1}^2\|r_{j+1} + y_{j+1} - y_0\|^2 \\ &\overset{(3.6)}{\leq} 4\theta m_{k+1}^2\left[\psi(y_0) - \psi(y_{j+1}) + \frac{1}{2}\|y_{j+1} - y_0\|^2\right] = 2\theta m_{k+1}\left[\phi(z_k) - \phi(z_{k+1})\right], \end{aligned}$$

which is exactly (2.3). Combing all previous inequalities yields the desired conclusion. ∎

Some remarks are in order. We first remark on Algorithm 3.1:

1. In view of (3.2), the number of iterations in its $(j+1)$-th call stops is bounded above by $1 + \log_\beta(L_{j+1}/L_j)$.
2. The update for y is equivalent to

$$y = \operatorname*{argmin}_{z \in \operatorname{dom}\psi^n}\left\{\frac{\psi^n(z)}{L+\mu} + \frac{1}{2}\left\|z - \left(\tilde{x} - \frac{\nabla\psi^s(\tilde{x})}{L+\mu}\right)\right\|^2\right\}$$

   which is a single call to the prox-oracle of $\psi^n/(L+\mu)$.
3. The descent condition (3.4) is well-known in existing literature for adaptive FISTA-type methods (see, for example, [41, Subsection 4.3]).

We now remark on Algorithm 3.2 and its associated results:

4. It is shown in Lemma 3.1 that (i) r_{j+1} is a stationarity residual for the iterate y_{j+1} and (ii) $\{L_j\}_{j\geq 0}$ forms a nondecreasing sequence of nonnegative scalars.
5. Step 1 is generally where most of the computation is done, wherein (possibly) multiple accelerated gradient steps are performed using Algorithm 3.1. It is also the only step that requires evaluating the prox oracle for ψ^n.
6. It is shown in Proposition 3.2(b) that both inequalities in Step 2 hold when ψ^s is μ-strongly convex. The first (resp. second) inequality of (3.5) is used to ensure that the first (resp. second) inequality of (3.6) holds when enough iterations are performed. See the analysis in Subsection 4.1 for more details.

7. Condition (3.6) is chosen so that Algorithm 3.2 implements a single step of Algorithm 2.1 if it stops in Step 3 and it is given the right inputs (see Lemma 3.3).
8. Suppose Algorithm 3.2 terminates in J iterations. Then, the number iterations of Algorithm 3.1 taken by Algorithm 3.2 is

$$\sum_{j=0}^{J-1}\left[1+\log_\beta \frac{L_{j+1}}{L_j}\right] = J + \log_\beta \frac{L_J}{L_0} \leq J + \log_\beta \frac{\overline{L}}{L_0}.$$

Thus, on average (up to a $\Theta(\log \overline{L}/J)$ additive term) Algorithm 3.2 uses only one accelerated gradient step or two function and prox oracle calls. It is worth mentioning that Nesterov's universal fast gradient method [36, Section 4] uses on average (up to a $\Theta(\log \overline{L}/J)$ additive term) four function/prox oracle calls per invocation.

3.2. PF.APD Algorithm. Broadly speaking, PF.APD is a *double-loop* method consisting of *outer iterations* and (possibly) several *inner iterations* per outer iteration. More specifically, the $(k+1)$-th outer iteration of PF.APD repeatedly applies Algorithm 3.2 to the proximal subproblem

$$z_{k+1} \approx \operatorname*{argmin}_{z \in \operatorname{dom} h} \left\{ \frac{\phi(z)}{2\hat{m}} + \frac{1}{2}\|z - z_k\|^2 \right\},$$

for increasing values of $\hat{m} > 0$, where z_k is an approximate solution to the k-th subproblem. On the other hand, the inner iterations refer to the iterations performed by Algorithm 3.2.

We now present the details of PF.APD and its key properties. To help our discussion, we first give the complete pseudocode of PF.APD through Algorithm 3.1 and Algorithm 3.4. More specifically, Algorithm 3.1 presents the (lower curvature) line search strategy used in PF.APD, while Algorithm 3.4 describes the other steps of PF.APD and how Algorithm 3.3 is invoked.

We next present three important properties about Algorithm 3.4 and its iterates. As its proof is mostly technical, we move it to Subsection 4.2. Moreover, to ensure that the resulting properties account for the possible asymmetry in (1.2), we make use of the scalars

$$\begin{aligned} m_* &:= \operatorname*{argmin}_{z,z' \in \operatorname{dom} h,\ t \geq 0} \left\{ t : f(z) - \ell_f(z; z') \geq -\frac{t}{2}\|z - z\|^2 \right\}, \\ M_* &:= \operatorname*{argmin}_{z,z' \in \operatorname{dom} h,\ t \geq 0} \left\{ t : f(z) - \ell_f(z; z') \leq \frac{t}{2}\|z - z\|^2 \right\}, \end{aligned} \tag{3.13}$$

which are the values of a curvature pair of f.

PROPOSITION 3.4. *Define the scalars*

$$\begin{aligned} \overline{m} &:= \max\{m_0, (\alpha+\beta)m_*\}, \quad \overline{M} := \beta\left[\max\{M_0, M_*\} + 2\overline{m}\right], \\ \overline{\mathcal{L}}_0 &:= \frac{\overline{M}}{2m_0} + 1, \quad P_0 := \log^{1+}\left\{\overline{\mathcal{L}}_0 \mathcal{A}_{\frac{1}{2},\overline{\mathcal{L}}_0}\left(\frac{1}{4}, \theta\right)\right\}, \end{aligned} \tag{3.14}$$

where (m_*, M_*) *and* $\mathcal{A}_{\mu,\bar{L}}(\cdot,\cdot)$ *are as in* (3.13) *and* (3.8)*, respectively. Then, for every* $k \geq 0$*, the following statements hold about* Algorithm 3.4 *and its iterates:*

Algorithm 3.3 Line Search and Proximal Descent Step

`Data`: (ψ^s, ψ^n, f, h) as in (3.3), $\hat{z} \in \operatorname{dom} h$, $\hat{m} \in \mathbb{R}_{++}$, $\hat{M} \in [m, \infty)$;
`Hyper-parameters`: $\theta \in (2, \infty)$, $\alpha \in (1, \infty)$, $\beta \in (1, \infty)$;
`Outputs`: $(z, u, m, M) \in \operatorname{dom} h \times \mathbb{R}^n$;

$M \leftarrow \hat{M}$
$\phi(\cdot) \leftarrow f(\cdot) + h(\cdot)$
for $\ell \leftarrow 0, 1, \ldots$ **do**
    $m \leftarrow \hat{m}\alpha^\ell$
    ▷ `Step 1:` $(\ell+1)^{\text{th}}$ `proximal subproblem.`
    call Algorithm 3.2 with data (ψ^s, ψ^n), $y_0 \equiv \hat{z}$, $\mu \equiv 1/2$, $L_0 \equiv M/(2m) + 1$, and hyper-parameters $\sigma \equiv 1/4$, θ, β, to obtain an output tuple (z, r, L)
    $u \leftarrow 2mr$
    $M \leftarrow 2m(L - 1)$
    ▷ `Step 2: Proximal descent check.`
    if the inequalities

$$\|u + 2m(z - \hat{z})\|^2 \le 2\theta m \left[\phi(\hat{z}) - \phi(z)\right], \qquad \|u\|^2 \le m^2 \|z - \hat{z}\|^2, \tag{3.11}$$

    hold, then **return** (z, u, m, M)

Algorithm 3.4 Parameter-Free Accelerated Proximal Descent (PF.APD) Algorithm

`Data`: (f, h) as in ⟨`A1`⟩–⟨`A3`⟩, $z_0 \in \operatorname{dom} h$, $m_0 \in \mathbb{R}_{++}$, $M_0 \in [m_0, \infty)$, $\varepsilon \in \mathbb{R}_{++}$;
`Hyper-parameters`: $\theta \in (2, \infty)$, $\alpha \in (1, \infty)$, $\beta \in (1, \infty)$;
`Outputs`: $(z_{k+1}, v_{k+1}) \in \operatorname{dom} h \times \mathbb{R}^n$;

for $k \leftarrow 0, 1, \ldots$ **do**
    ▷ `Step 1: Line search for` m_{k+1} `and proximal descent step.`

$$\hat{m} \leftarrow \begin{cases} m_k/\alpha, & \text{if } k \ge 1 \text{ and } m_k < \cdots < m_0, \\ m_k, & \text{otherwise} \end{cases}$$

    call Algorithm 3.3 with data

$$\psi^s(\cdot) = \frac{f(\cdot)}{2\hat{m}} + \frac{1}{2}\|\cdot - z_k\|^2, \quad \psi^n(\cdot) = \frac{h(\cdot)}{2\hat{m}}, \tag{3.12}$$

    (f, h), $\hat{z} \equiv z_k$, $\hat{m} \equiv \hat{m}$, $\hat{M} \equiv M_k$, and hyper-parameters θ, α, β to obtain $(z_{k+1}, u_{k+1}, m_{k+1}, M_{k+1})$
    ▷ `Step 2: Stationarity termination check.`
    $v_{k+1} \leftarrow 2m_{k+1}(u_{k+1} + z_k - z_{k+1})$
    if $\|v_{k+1}\| \le \varepsilon$ **then**
        return $(z_{k+1},\ v_{k+1})$

(a) $M_k \le M_{k+1} \le \overline{M} < \infty$ and $\{1/m_j\}$ is bitonic[4] and bounded below by $1/\overline{m}$;
(b) its $(k+1)$-th outer iteration performs at most T_{k+1} inner iterations, where

$$T_{k+1} \le 20 \left(1 + \log_\alpha \frac{m_{k+1}}{m_k} + \frac{1}{\sqrt{\alpha} - 1}\sqrt{\frac{\overline{M}}{2m_k}}\right) P_0; \tag{3.15}$$

[4] A sequence $\{a_k\}_{k=0}^n$ is *bitonic* if there exists $0 \le j < n$ such that $a_0 \le \cdots \le a_j \ge \cdots \ge a_n$. Note that monotone sequences are bitonic as well.

(c) it performs a finite number of outer iterations $K(\varepsilon)$, where

$$K(\varepsilon) \le 1 + \sum_{k=0}^{K(\varepsilon)-2} \frac{\overline{m}}{m_{k+1}} < 1 + \frac{2\theta\Delta_0\overline{m}}{\varepsilon^2}; \tag{3.16}$$

(d) if, in addition, $m_0 \ge m_*$, then $m_j = \alpha^{-j} m_0$ for every $j \ge 0$ and $K(\varepsilon)$ in (3.16) also satisfies

$$K(\varepsilon) \le 2 + \log_\alpha \left[\alpha^4 + \frac{2\theta m_0^2 R_{m_0}(z_0)}{\varepsilon^2}\right], \tag{3.17}$$

where $R_\tau(\cdot)$ is as in (2.5);

(e) $v_{k+1} \in \nabla f(z_{k+1}) + \partial h(z_{k+1})$ and its final iterate $(\bar{z}, \bar{v}) = (z_{k+1}, v_{k+1})$ solves Problem $\boldsymbol{\mathcal{CO}}$.

We are now ready to give some important iteration complexity bounds on Algorithm 3.4.

THEOREM 3.5. *Define $Q_0 := 20P_0\left[1 + \log_\alpha(\overline{m}/m_0)\right]$, where $\overline{m}$ and P_0 are as in (3.14), respectively. Then, Algorithm 3.4 stops and outputs a pair $(\bar{z}, \bar{v}) = (z_{k+1}, v_{k+1})$ solving Problem $\boldsymbol{\mathcal{CO}}$ in $\overline{T}$ inner iterations, where*

$$\overline{T} \le Q_0 + \frac{20P_0}{\sqrt{\alpha}-1}\sqrt{\overline{M}\left[1 + \frac{2\theta\Delta_0\overline{m}}{\varepsilon^2}\right]\left[\frac{1}{m_0} + \frac{2\theta\Delta_0}{\varepsilon^2}\right]}, \tag{3.18}$$

and Δ_0 is as in (1.8). Moreover, if $m_0 \ge m_*$, then

$$\overline{T} \le Q_0 + \frac{20P_0\alpha}{(\sqrt{\alpha}-1)^2}\left[\frac{\alpha^2}{\sqrt{m_0}} + \frac{\sqrt{\theta \min\{2\Delta_0, m_0 d_0^2\}}}{\varepsilon}\right], \tag{3.19}$$

where d_0 is as in (1.8).

Proof. The fact that Algorithm 3.4 stops in a finite number of inner iterations with a pair solving Problem $\mathcal{CO}$ is immediate from Proposition 3.4. Furthermore, the previous proposition also implies that the total number of inner iterations in a single call of Algorithm 3.4 is at most

$$\begin{aligned}
\sum_{k=0}^{K(\varepsilon)-1} T_{k+1} &\le 20P_0 \sum_{k=0}^{K(\varepsilon)-1}\left(1 + \log_\alpha \frac{m_{k+1}}{m_k} + \frac{1}{\sqrt{\alpha}-1}\sqrt{\frac{\overline{M}}{2m_k}}\right) \\
&\le 20P_0\left(1 + \log_\alpha \frac{m_{K(\varepsilon)+1}}{m_0} + \frac{\sqrt{\overline{M}}}{\sqrt{\alpha}-1}\sum_{k=0}^{K(\varepsilon)-1}\frac{1}{\sqrt{m_k}}\right) \\
&\le Q_0 + \frac{20P_0\sqrt{\overline{M}}}{\sqrt{\alpha}-1}\sum_{k=0}^{K(\varepsilon)-1}\frac{1}{\sqrt{m_k}}, \qquad (3.20)
\end{aligned}$$

where T_{k+1} and $K(\varepsilon)$ are as in (3.15) and (3.16), respectively. Let us now bound the sum $\sum_{k=0}^{K(\varepsilon)-1} m_k^{-1/2}$. Using Proposition 3.4(c) and the fact that $\|z\|_1 \le \sqrt{n}\|z\|_2$ for any $z \in \mathbb{R}^n$, we first have

$$\sum_{k=0}^{K(\varepsilon)-1}\frac{1}{\sqrt{m_k}} \le \left[K(\varepsilon)\sum_{k=0}^{K(\varepsilon)-1}\frac{1}{m_k}\right]^{1/2} \le \sqrt{\left(1 + \frac{2\theta\Delta_0\overline{m}}{\varepsilon^2}\right)\left(\frac{1}{m_0} + \frac{2\theta\Delta_0}{\varepsilon^2}\right)}.$$

Using (3.20) and the above bound yields (3.18).

Now, let $\mathcal{R}_0 := R_{m_0}(z_0)$ and suppose $m_0 \geq m_*$. Using Proposition 3.4(d), (2.9) with $\tilde{m} = m_0\alpha^{-K(\varepsilon)}$ and $\nu = \tilde{m}/m_0$, and the inequality $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ for $a, b \in \mathbb{R}$, we have

$$
\begin{aligned}
\sum_{k=0}^{K(\varepsilon)-1} \frac{1}{\sqrt{m_k}} &= \sum_{k=0}^{K(\varepsilon)-1} \sqrt{\frac{\alpha^k}{m_0}} \leq \frac{\alpha^{K(\varepsilon)/2}}{\sqrt{m_0}(\sqrt{\alpha}-1)} \\
&\overset{(d)}{\leq} \frac{\alpha}{\sqrt{m_0}(\sqrt{\alpha}-1)} \sqrt{\alpha^4 + \frac{2\theta m_0^2 R_{m_0}(z_0)}{\varepsilon^2}} \leq \frac{\alpha}{(\sqrt{\alpha}-1)\varepsilon}\left[\frac{\alpha^2}{\sqrt{m_0}} + \frac{\sqrt{2\theta m_0 \mathcal{R}_0}}{\varepsilon}\right] \\
&\overset{(2.9)}{\leq} \frac{\alpha}{(\sqrt{\alpha}-1)\varepsilon}\left[\frac{\alpha^2}{\sqrt{m_0}} + \frac{\sqrt{\theta \min\{2\Delta_0, m_0 d_0^2\}}}{\varepsilon}\right].
\end{aligned}
$$

Combining (3.20) and the above bound yields (3.19). ∎

Some remarks are in order. We first remark on Algorithm 3.3:

1. In view of assumption ⟨A2⟩ and Proposition 3.2, the number of iterations in its k-th call is bounded above by $1 + \log_\alpha(m_{k+1}/m_k)$.
2. The checks in its Step 2 correspond to (2.3) and (2.4), respectively.
3. If the ℓ-th call to Algorithm 3.2 ends with a "bad termination", i.e., Step 2 in Algorithm 3.2, then (3.11) does not hold, the estimate m is increased by a factor of α, and the algorithm proceeds to the $(\ell+1)$-th iteration.

We now remark on Algorithm 3.4 and its associated results:

4. It is shown in Proposition 3.4 that (i) v_{j+1} is a stationarity residual for the iterate z_{j+1} and (ii) $\{M_k\}_{k\geq 0}$ and $\{m_k\}_{k\geq 0}$ are nondecreasing and nonnegative.
5. Q_0 in (3.18)–(3.19) bounds the total number of inner iterations performed by unsuccessful calls to Algorithm 3.2, i.e., those that stop in Step 2 of Algorithm 3.2.
6. While m_0 and M_0 are free parameters, a good initial value[5] for them is an estimate of the local Lipchitz constant $\tilde{L}_0$ of ∇f at z_0. Similar to the approach in [32], one can estimate $\tilde{L}_0$ by sampling some $\hat{z} \in \operatorname{dom} h$ with $\hat{z} \neq z_0$ and choosing $\tilde{L}_0 = \|\nabla f(z_0) - \nabla f(\hat{z})\|/\|z_0 - \hat{z}\|$.

Before ending the section, we discuss how different choices of m_0 affect the complexities in (3.18) and (3.19) when $m_* \leq M_*$:

7. In the general case, choosing $m_0 = 1$ implies that the bound in (3.18) (resp. (3.19)) is $\mathcal{O}(\sqrt{M_* m_*}\Delta_0/\varepsilon^2)$ (resp. $\mathcal{O}(\sqrt{M_*\Delta_0}/\varepsilon)$) which matches the complexity of the AIPP in [18] and is optimal[6] for finding stationary points of (1) in the weakly-convex (resp. convex) setting in terms of of m_*, M_*, Δ_0, and ε.
8. If d_0 is known, then choosing $m_0 = \varepsilon/d_0$ implies (3.19) is $\tilde{\mathcal{O}}(\sqrt{M_* d_0}/\sqrt{\varepsilon})$ which is optimal[7], up to logarithmic terms, for finding stationary points of (1) in the convex setting in terms of of M_*, d_0, and ε.

4. Technical Proofs.

This section gives the proofs of several technical results in Section 3. More specifically, it presents the proofs of Lemma 3.1 and Proposition 3.4.

4.1. Proof of Lemma 3.1.

To avoid repetition, we let

$$\{(A_j, \tilde{x}_j, y_j, x_j, L_j)\}_{j\geq 0} \tag{4.1}$$

[5] This is motivated by the fact that m_0 and M_0 are bounded by the Lipchitz constant of ∇f.

[6] See [48, Theorem 4.7].

[7] See [37, Section 2.2.2] or [7, Theorem 1].

denote the sequence of iterates generated by a single call to Algorithm 3.2 and define

$$
\begin{gathered}
a_i := A_{i+1} - A_i, \quad \xi_i := 1 + \mu A_i, \\
\tilde{q}_{i+1}(\cdot) := \ell_{\psi^s}(\cdot; \tilde{x}_i) + \psi^n(\cdot) + \frac{\mu}{2}\|\cdot - \tilde{x}_i\|^2, \\
q_{i+1}(\cdot) := \tilde{q}_{i+1}(y_{i+1}) + L_{i+1}\langle \tilde{x}_i - y_{i+1}, \cdot - y_{i+1}\rangle + \frac{\mu}{2}\|\cdot - y_{i+1}\|^2,
\end{gathered}
$$

for every $i \geq 0$. Recall also that each iterate in (4.1) is obtained in a finite number of iterations of Algorithm 3.1 in view of (3.2) and (3.4).

We first present some basic technical properties about $\tilde{q}$ and q.

Lemma 4.1. *If ψ^s is μ-strongly convex, then, for every $j \geq 0$,*
(a) $\tilde{q}_{j+1}(y_{j+1}) = q_{j+1}(y_{j+1})$ and $\tilde{q}_{j+1}(\cdot) \leq q_{j+1}(\cdot) \leq \psi(\cdot)$;
(b) $y_{j+1} = \min_{x\in\mathbb{R}^n}\left\{q_{j+1}(x) + L_{j+1}\|x - \tilde{x}_{j+1}\|^2/2\right\}$;
(c) $x_{j+1} = \operatorname{argmin}_{x\in\mathbb{R}^n}\left\{a_j q_{j+1}(x) + \xi_{j+1}\|x - x_j\|^2/2\right\}$.

Proof. (a) See [17, Lemma B.0.1].

(b) Let $\Psi(\cdot) = q_{j+1}(\cdot) + L_{j+1}\|\cdot - \tilde{x}_{j+1}\|^2/2$. It follows from the definition of q_{j+1} that $\nabla\Psi(y_{j+1}) = 0$ and, hence, y_{j+1} satisfies the optimality condition of the given inclusion.

(c) Using the definition of q_{j+1}, the given optimality condition of x_{j+1} holds if and only if

$$
x_{j+1} = x_j - \frac{a_j \nabla q_{j+1}(x_j)}{\xi_{j+1}} = x_j + \frac{a_j\left[L(y_{j+1} - \tilde{x}_j) + \mu(y_{j+1} - x_j)\right]}{1 + \mu A_{j+1}}
$$

which is equivalent to the update for x_{j+1} in Algorithm 3.2 (given by Algorithm 3.1). ∎

The next result presents an important technical bound on the residual $\|y_{j+1} - \tilde{x}_j\|^2$.

Lemma 4.2. *If ψ^s is μ-strongly convex, then, for every $j \geq 0$ and $y \in \mathbb{R}^n$,*

$$
\begin{aligned}
&\frac{\mu A_{j+1}}{2}\|y_{j+1} - \tilde{x}_j\|^2 + A_{j+1}\psi(y_{j+1}) + \frac{\xi_{j+1}}{2}\|y - x_{j+1}\|^2 \\
&\qquad \leq A_j q_{j+1}(y_j) + a_j q_{j+1}(y) + \frac{\xi_j}{2}\|y - x_j\|^2.
\end{aligned} \tag{4.2}
$$

Proof. Let $y \in \mathbb{R}^n$ be fixed. We first derive two auxiliary technical inequalities. For the first one, we use the fact that $a_j q_{j+1} + \xi_j\|\cdot - x_j\|^2/2$ is ξ_{j+1}-strongly convex, the definition of ξ_{j+1}, and the optimality of x_{j+1} in Lemma 4.1(c) to obtain

$$
a_j q_{j+1}(y) + \frac{\xi_j}{2}\|y - x_j\|^2 - \frac{\xi_{j+1}}{2}\|y - x_{j+1}\|^2 \geq a_j q_{j+1}(x_{j+1}) + \frac{\xi_j}{2}\|x_{j+1} - x_j\|^2. \tag{4.3}
$$

For the second one, let $r_{j+1} := (A_j y_j + a_j x_{j+1})/A_{j+1}$. Using the convexity of q_{j+1}, the updates in Algorithm 3.1 and Algorithm 3.2, and Lemma 4.1(a)–(b), we obtain

$$
\begin{aligned}
&A_j q_{j+1}(y_j) + a_j q_{j+1}(x_{j+1}) + \frac{\xi_j}{2}\|x_{j+1} - x_j\|^2 \\
&\geq A_{j+1}\left[q_{j+1}(r_{j+1}) + \frac{\xi_j}{2a_j^2}\left\|r_{j+1} - \frac{A_j y_j + a_j x_j}{A_{j+1}}\right\|^2\right] \\
&= A_{j+1}\left[q_{j+1}(r_{j+1}) + \frac{L_{j+1}}{2}\|r_{j+1} - \tilde{x}_j\|^2\right] \geq A_{j+1}\min_{x\in\mathbb{R}^n}\left\{q_{j+1}(x) + \frac{L_{j+1}}{2}\|x - \tilde{x}_j\|^2\right\} \\
&\overset{\text{Lemma 4.1(a)-(b)}}{=} A_{j+1}\left[\tilde{q}_{j+1}(y_{j+1}) + \frac{L_{j+1}}{2}\|y_{j+1} - \tilde{x}_j\|^2\right]
\end{aligned} \tag{4.4}
$$

Combining (4.3), (4.4), and (3.4) with $L = L_{j+1}$, we conclude that

$$
\begin{aligned}
&A_j q_{j+1}(y_j) + a_j q_{j+1}(y) + \frac{\xi_j}{2}\|y - x_j\|^2 - \frac{\xi_{j+1}}{2}\|y - x_{j+1}\|^2 \\
&\overset{(4.3)}{\geq} A_j q_{j+1}(y_j) + a_j q_{j+1}(x_{j+1}) + \frac{\xi_j}{2}\|x_{j+1} - x_j\|^2 \\
&\overset{(4.4)}{\geq} \tilde{q}_{j+1}(y_{j+1}) + \frac{L_{j+1}}{2}\|y_{j+1} - \tilde{x}_j\|^2 \overset{(3.4)}{\geq} \psi(y_{j+1}) + \frac{\mu}{2}\|y_{j+1} - \tilde{x}_j\|^2. \qquad \square
\end{aligned}
$$

The following result further refines the previous bound on $\|y_{j+1} - \tilde{x}_j\|^2$.

LEMMA 4.3. *If ψ^s is μ-strongly convex, then, for every $j \geq 0$,*

$$
\mu A_{j+1}\|y_{j+1} - \tilde{x}_j\|^2 \leq \|y_{j+1} - y_0\|^2 - \xi_{j+1}\|y_{j+1} - x_{j+1}\|^2. \tag{4.5}
$$

Proof. Let $j \geq 0$ be fixed and suppose ψ^s is μ-strongly convex. Moreover, define

$$
\Psi_i := A_i \left[\psi(y_i) - \psi(y_j)\right] + \frac{\xi_i}{2}\|y_j - x_i\|^2 \quad \forall i \geq 0.
$$

Using Lemma 4.2 with $y = y_j$, Lemma 4.1(a), the fact that $a_j = A_{j+1} - A_j$, and the definition of Ψ_i above, we have that for every $i \geq 0$,

$$
\begin{aligned}
&\frac{\mu A_{i+1}}{2}\|y_{i+1} - \tilde{x}_i\|^2 \\
&\overset{(4.2)}{\leq} A_i q_{i+1}(y_i) + a_i q_{i+1}(y_j) + \frac{\xi_i}{2}\|y_i - x_i\|^2 - \Psi_{i+1} - A_{i+1}\psi(y_j) \\
&\overset{\textit{Lemma } 4.1\text{(a)}}{\leq} A_i \psi(y_i) + a_i \psi(y_j) + \frac{\xi_i}{2}\|y_i - x_i\|^2 - \Psi_{i+1} - A_{i+1}\psi(y_j) \\
&= \Psi_i - \Psi_{i+1}.
\end{aligned}
$$

Summing the above inequality from $i = 0$ to j and using the fact that $A_{i+1} \geq 0$ for every i and $(x_0, A_0, \xi_0) = (y_0, 0, 1)$, we conclude that

$$
\begin{aligned}
&\frac{\mu A_{j+1}}{2}\|y_{j+1} - \tilde{x}_j\|^2 \leq \sum_{i=0}^{j} \frac{\mu A_{i+1}}{2}\|y_{i+1} - \tilde{x}_i\|^2 \leq \Psi_0 - \Psi_{j+1} \\
&= \frac{\xi_0}{2}\|y_j - x_0\|^2 - \frac{\xi_j}{2}\|y_{j+1} - x_{j+1}\|^2 = \frac{1}{2}\|y_j - y_0\|^2 - \frac{\xi_j}{2}\|y_{j+1} - x_{j+1}\|^2. \qquad \square
\end{aligned}
$$

We are now ready to prove Lemma 3.1.

Proof of Lemma 3.1. (a) See [17, Lemma B.0.2] for the bound on A_{j+1}. The bound on L_j follows from how Algorithm 3.1 is called in Algorithm 3.2, the update rule for L in Algorithm 3.1, and (3.2) which follows from assumption $\langle$B2$\rangle$.

(b) Using the optimality of y_{j+1} given by Algorithm 3.1 and Algorithm 3.2 and the definition of r_{j+1}, it follows that

$$
0 \in \nabla\psi^s(\tilde{x}_j) + \partial\psi^n(y_{j+1}) + (L_{j+1} + \mu)(y_{j+1} - \tilde{x}_j) = \nabla\psi^s(y_{j+1}) + \partial\psi^n(y_{j+1}) - r_{j+1}.
$$

(c) The first bound in (3.4) is an immediate consequence of Lemma 4.3. For the second bound in (3.4), note that part (b) and the assumption that ψ^s implies that $r_{j+1} \in \partial\psi(y_{j+1})$. The conclusion now follows from the previous inclusion and the definition of the subdifferential.

(d) Suppose $A_{j+1} \geq \mathcal{A}_{\mu,\bar{L}} := \mathcal{A}_{\mu,\bar{L}}(\sigma,\theta)$ and (3.4) holds. We separate this proof into two parts. We first prove the bound in (3.4). Using the definitions of r_{j+1} and $\bar{L}$, part (c), the fact that $\mu \leq L_0 \leq L_{j+1}$, assumption ⟨B2⟩, and the relation $(a+b)^2 \leq 2a^2 + 2b^2$ for $a, b \in \mathbb{R}$, we have that

$$\begin{aligned}
\|r_{j+1}\|^2 &= \|\nabla\psi^s(y_{j+1}) - \nabla\psi^s(\tilde{x}_j) + (L_{j+1}+\mu)(\tilde{x}_j - y_{j+1})\|^2 \\
&\leq 2\|\nabla\psi^s(y_{j+1}) - \nabla\psi^s(\tilde{x}_j)\|^2 + 2(L_{j+1}+\mu)^2\|\tilde{x}_j - y_{j+1}\|^2 \\
&\leq 2[L_*^2 + (L_{j+1}+\mu)^2]\|\tilde{x}_j - y_{j+1}\| \leq 16\bar{L}^2\|\tilde{x}_j - y_{j+1}\|^2 \\
&\overset{(4.5)}{\leq} \frac{16\bar{L}}{\mu A_{j+1}}\|y_{j+1} - y_0\|^2.
\end{aligned}$$

It follows from the above bound and the definition of $\mathcal{A}_{\mu,\bar{L}}$ that

$$\|r_{j+1}\|^2 \leq \frac{16\bar{L}}{\mu A_{j+1}}\|y_{j+1} - y_0\|^2 \leq \frac{16\bar{L}^2}{\mu \mathcal{A}_{\mu,\bar{L}}}\|y_{j+1} - y_0\|^2 \leq \sigma^2\|y_{j+1} - y_0\|^2$$

and, hence, the first condition of (3.6) holds.

To show the second condition of (3.6), let $\gamma := \sqrt{(2-\theta)/\theta}$. Using the fact that $\gamma \in (0,1)$, (3.4), $\mu \leq L_{j+1}$, and the bound

$$\|a+b\|^2 \leq (1+\gamma)\|a\|^2 + (1+\gamma^{-1})\|b\|^2 \quad \forall a, b \in \mathbb{R}^n,$$

we then have that

$$\begin{aligned}
&\|r_{j+1}\|^2 \overset{(3.4)}{\leq} \frac{L^2}{\mu A_{j+1}}\|y_{j+1} - y_0\|^2 \leq \frac{4(\mu + L_{j+1})^2}{\mu A_{j+1}}\|y_{j+1} - y_0\|^2 \\
&\leq \frac{16\bar{L}^2}{\mu \mathcal{A}_{\mu,\bar{L}}(\sigma,\theta)}\|y_{j+1} - y_0\|^2 \leq \frac{\gamma^2}{4}\|y_{j+1} - y_0\|^2 \overset{\gamma\in(0,1)}{\leq} \left(\frac{\gamma}{1+\gamma}\right)^2\|y_{j+1} - y_0\|^2 \\
&\leq \left(\frac{\gamma}{1+\gamma}\right)^2(1+\gamma)\|r_{j+1} + y_{j+1} - y_0\|^2 + \left(\frac{\gamma}{1+\gamma}\right)^2\left(1 + \frac{1}{\gamma}\right)\|r_{j+1}\|^2 \\
&= \frac{\gamma^2}{1+\gamma}\|r_{j+1} + y_{j+1} - y_0\|^2 + \frac{\gamma}{1+\gamma}\|r_{j+1}\|^2,
\end{aligned}$$

which implies $\|r_{j+1}\|^2 \leq \gamma^2\|r_{j+1} + y_{j+1} - y_0\|^2$. It then follows from the second bound in (3.4) and the previous inequality that

$$\begin{aligned}
2[\psi(y_0) - \psi(y_{j+1})] &\overset{(3.5)}{\geq} 2\langle r_{j+1}, y_0 - y_{j+1}\rangle \\
&= \|r_{j+1} + y_0 - y_{j+1}\|^2 - \|r_{j+1}\|^2 - \|y_0 - y_{j+1}\|^2 \\
&\geq (1-\gamma^2)\|r_{j+1} + y_0 - y_{j+1}\|^2 - \|y_0 - y_{j+1}\|^2 \\
&= \frac{2}{\theta}\|r_{j+1} + y_0 - y_{j+1}\|^2 - \|y_0 - y_{j+1}\|^2. \qquad \square
\end{aligned}$$

4.2. Proof of Proposition 3.4.

Proof of Proposition 3.4. (a) Note that the k-th successful call of Algorithm 3.2 is such that its input ψ^s has the curvature pair

$$(L_{k+1}^-, L_{k+1}^+) := \left(\max\left\{0, \frac{m_*}{2m_{k+1}} - 1\right\}, \frac{M_*}{2m_{k+1}} + 1\right). \tag{4.6}$$

Hence, it follows from Step 1 of Algorithm 3.3, Proposition 3.2(b) with $\mu = 1/2$, and the definition of $\overline{m}$ imply that the last call of Algorithm 3.2 at the k-th iteration of Algorithm 3.4 obtains m_{k+1} being at most $\alpha m_k \le \overline{m}$. Consequently, $\{1/m_k\}$ (resp. $\{m_k\}$) is bounded below by $1/\overline{m}$ (resp. bounded above by $\overline{m}$). The fact that $\{1/m_j\}$ is bitonic follows from the the definition of $\hat{m}$ in Step 1 of Algorithm 3.4, the call to Algorithm 3.3 in of Algorithm 3.4, and the fact that in Algorithm 3.4 the returned scalar m in is always lower bounded by the input $\hat{m}$. To show the bound on M_k, note that the curvature pair of ψ^s in (4.6) implies that $\nabla\psi^s$ is L_*-Lipschitz continuous where $L_* = \max\{L_{k+1}^-, L_{k+1}^+\}$. It then follows from the upper previous bound on m_{k+1} and Lemma 3.1(a) that

$$\begin{aligned}\frac{M_k}{2m_{k+1}} + 1 &\le \frac{M_{k+1}}{2m_{k+1}} + 1 \le \beta\left[\frac{\max\{M_0, M_*\}}{2m_{k+1}} + 1\right]\\ &\le \frac{\beta\left[\max\{M_0, M_*\} + 2\overline{m}\right]}{2m_{k+1}} = \frac{\overline{M}}{2m_{k+1}},\end{aligned}$$

which immediately implies $M_{k+1} \ge M_k$ and $M_{k+1} \le \overline{M}$.

(b) Let an outer iteration index $k \ge 1$ be fixed and define

$$\mathcal{L}_\ell := \frac{\overline{M}}{2m_k\alpha^\ell} + 1, \quad \mathcal{I}_\ell := \left\lceil 1 + 4\sqrt{\mathcal{L}_\ell}P_0\right\rceil, \quad \bar{\ell} := 1 + \log_\alpha(m_{k+1}/m_k),$$

where P_0 is as in (3.14). Using Proposition 3.2(a) with $(\mu, \sigma) = (1/2, 1/4)$, part (a), the fact that $P_0 \ge 1$, and assumptions ⟨A1⟩–⟨A2⟩, it follows that the number of inner iterations performed by Algorithm 3.4 at outer iteration k is bounded above by

$$\begin{aligned}\sum_{\ell=0}^{\bar{\ell}}\mathcal{I}_\ell &\le 2\sum_{\ell=0}^{\bar{\ell}}\left(1 + 4\sqrt{\mathcal{L}_\ell}P_0\right) \le 2\sum_{\ell=0}^{\bar{\ell}}\left(1 + 4\left[\sqrt{\frac{\overline{M}}{2m_k\alpha^\ell}} + 1\right]P_0\right)\\ &\le 10P_0\sum_{\ell=0}^{\bar{\ell}}\left(\sqrt{\frac{\overline{M}}{2m_k\alpha^\ell}} + 1\right) = 10\left[\bar{\ell} + \sqrt{\frac{\overline{M}}{2m_k}}\sum_{\ell=0}^{\bar{\ell}}\alpha^{-\ell/2}\right]P_0\\ &= 10\left[\bar{\ell} + \sqrt{\frac{\overline{M}}{2m_k}}\left(\frac{1 - \alpha^{-\bar{\ell}/2}}{\sqrt{\alpha} - 1}\right)\right]P_0 \le 10\left[\bar{\ell} + \frac{1}{\sqrt{\alpha} - 1}\sqrt{\frac{\overline{M}}{2m_k}}\right]P_0\\ &\le 20\left[1 + \log_\alpha\frac{m_{k+1}}{m_k} + \frac{1}{\sqrt{\alpha} - 1}\sqrt{\frac{\overline{M}}{2m_k}}\right]P_0.\end{aligned}$$

(c) In view of Proposition 3.4(a), let $\overline{K}$ be an index satisfying

$$\frac{\overline{K} - 1}{\overline{m}} \le \sum_{k=0}^{\overline{K}-2}\frac{1}{m_{k+1}} < \frac{2\theta\Delta_0}{\varepsilon^2} \le \sum_{k=0}^{\overline{K}-1}\frac{1}{m_{k+1}}.$$

Using Lemma 3.3, the choice of inputs to Algorithm 3.2, and Lemma 2.1(b), and the last of the above inequalities, we have that

$$\inf_{0\le j\le\overline{K}-1}\|v_{j+1}\|^2 \le \frac{2\theta\Delta_0}{\sum_{k=0}^{\overline{K}-1}m_{k+1}^{-1}} \le \varepsilon^2.$$

Hence, because of the termination condition in Step 2 of Algorithm 3.4, it follows that the number of outer iterations $K(\varepsilon)$ is at most $\overline{K}$. Using the fact that $m_{k+1} > 0$ for every $k \geq 0$, the bounds in (3.16) immediately follow.

(d) Suppose $m_0 \geq m_*$. Since f is m_*-weakly convex (see (3.13)), it follows from Proposition 3.2(b) and the choice of ψ^s in Algorithm 3.4 and μ in Algorithm 3.3 that Algorithm 3.1 is called only once at every (outer) iteration of Algorithm 3.4. Consequently, it follows from the definition of $\hat{m}$ in Step 1 of Algorithm 3.4 and part (a) that $m_j = \alpha^{-j} m_0$ for every $j \geq 0$. Now, in view of the fact that $\{1/m_j\}$ is bounded below from part (a), let $\overline{K}$ be the smallest index such that $\overline{K} \geq 2$ and

$$\sum_{k=1}^{\overline{K}-1} \frac{1}{m_{k+1}} \leq \frac{\alpha^2}{m_0} + \frac{2\theta m_0 R_{m_0}(z_0)}{\varepsilon^2} \leq \sum_{k=1}^{\overline{K}} \frac{1}{m_{k+1}} \tag{4.7}$$

Using the fact that $\{m_j\}$ is nonincreasing, (2.7) with $\tilde{m} = m_0 \alpha^{-\overline{K}}$ and $\nu = m_0/\tilde{m}$, the identity $m_j = \alpha^{-j} m_0$, and the same type of arguments as in part (c), we have that

$$\min_{1 \leq j \leq \bar{K}-1} \|v_{j+1}\|^2 \leq \frac{2\theta \nu \tilde{m} R_{\nu\tilde{m}}(z_0)}{\sum_{k=1}^{\overline{K}-1} m_{k+1}^{-1}} = \frac{2\theta m_0 R_{m_0}(z_0)}{-\alpha^2 m_0^{-1} + \sum_{k=1}^{\overline{K}} m_{k+1}^{-1}} \leq \varepsilon^2,$$

and, hence, the number of outer iterations $K(\varepsilon)$ is bounded above by $\overline{K}$. It now remains to show that $\overline{K}$ is bounded above by the expression on the right-hand side of (3.17). Using the identity $m_j = \alpha^{-j} m_0$ and the right-hand side of (4.7), we have

$$\frac{\alpha^2}{m_0} + \frac{2\theta m_0 R_{m_0}(z_0)}{\varepsilon^2} \geq \sum_{k=1}^{\overline{K}-1} \frac{1}{m_{k+1}} = \frac{\alpha^2}{m_0} \sum_{k=0}^{\overline{K}-2} \alpha^k \geq \frac{\alpha^{\overline{K}-1}}{m_0(\alpha-1)} \geq \frac{\alpha^{\overline{K}-2}}{m_0}.$$

Applying the function $\log_\alpha(\cdot)$ to both sides of the above inequality and re-arranging terms yields the desired bound on $\overline{K}$.

(e) Using the definition of v_{k+1} and Lemma 3.3 with ψ^s as in (3.12), we have

$$\begin{aligned} v_{k+1} &\in 2m_{k+1}\left[\nabla \psi^s(z_{k+1}) + \partial \psi^s(z_{k+1})\right] + 2m_{k+1}(z_k - z_{k+1}) \\ &= 2m_{k+1}\left[\frac{\nabla f(z_{k+1})}{2m_{k+1}} + (z_{k+1} - z_k) + \frac{\partial h(z_{k+1})}{m_{k+1}}\right] + 2m_{k+1}(z_k - z_{k+1}) \\ &= \nabla f(z_{k+1}) + \partial h(z_{k+1}). \end{aligned}$$

The fact that the last iterate solves Problem $\mathcal{CO}$ follows from the above inclusion and the termination condition in Step 2 of Algorithm 3.4. ∎

5. Applications. This section describes a few possible applications of Algorithm 3.4 in more general optimization frameworks.

Min-Max Smoothing. In [22], a smoothing framework was proposed for finding ε-stationary points of the nonconvex-concave min-max problem

$$\min_{x \in \mathbb{R}^n} \max_{y \in \mathbb{R}^l} \left[\phi(x, y) + h(x)\right] \tag{5.1}$$

where h is as in assumption ⟨A1⟩, $\phi(\cdot, y)$ is m_x-weakly convex and differentiable, $-\phi(x, \cdot)$ is proper closed convex, and $\nabla_x \phi(\cdot, \cdot)$ is Lipschitz continuous.

The framework considers finding an ε-stationary point of h plus a smooth approximation $\hat{p}$ of $\max_{y \in Y} \phi(\cdot, y)$. Choosing a special smoothing constant such that the curvature pair $(\hat{m}, \hat{M})$ of $\hat{p}$ satisfies $\hat{m} = m_x$ and $\hat{M} = \Theta(\varepsilon^{-1} D_y)$ (resp. $\hat{M} = \Theta(D_y^2 \varepsilon^{-2})$), where D_y is diameter of $\mathrm{dom}(-\phi(x,\cdot))$, it was shown that an ε-stationary point of $\hat{p}$ yields an ε-primal-dual (resp. directional) stationary point of (5.1).

If we use PF.APD with $m_0 = \varepsilon$ to obtain an ε-stationary point of $\hat{p}$ as above, then an ε-primal-dual (resp. directional) stationary point of (5.1) is obtained in $\tilde{\mathcal{O}}(\varepsilon^{-2.5})$ (resp. $\tilde{\mathcal{O}}(\varepsilon^{-3})$) inner iterations, and this matches, up to logarithmic terms, the complexity bounds for the smoothing method in [22]. Moreover, when $\phi(\cdot, y)$ is convex, the above complexity is $\tilde{\mathcal{O}}(\varepsilon^{-1})$ (resp. $\tilde{\mathcal{O}}(\varepsilon^{-1.5})$), and this appears to be the first parameter-free approach that could be used for min-max optimization. This approach also has the strong advantage that it does not need to know D_y.

Penalty Method. In [19], a penalty method is proposed for finding ε-KKT points of the linearly-constrained nonconvex optimization problem

$$\min_{x\in\mathbb{R}^n} \{\phi(x) := f(x) + h(x) : Ax = b\} \tag{5.2}$$

where (f, h) are as in $\langle$A1$\rangle$–$\langle$A3$\rangle$. It was shown that if the penalty method uses an algorithm $\mathcal{A}$ that needs $\mathcal{O}(T_{m,M}(\varepsilon))$ iterations to obtain an ε-stationary point of ϕ, then the total number of inner iterations of the penalty method (for finding an ε-KKT point) is $\tilde{\mathcal{O}}(T_{m,\varepsilon^{-2}}(\varepsilon))$.

If we use the PF.APD with $m_0 = \varepsilon$ as algorithm $\mathcal{A}$ above, then an ε-KKT point of (5.2) is obtained in $\tilde{\mathcal{O}}(\varepsilon^{-3})$ inner iterations which matches the complexity bound for the particular penalty method in [19] (which uses the AIPP in [18] for algorithm $\mathcal{A}$). Moreover, when f is convex, the above complexity is $\tilde{\mathcal{O}}(\varepsilon^{-1.5})$. Like in the above discussion for min-max smoothing, this appears to be the first parameter-free approach used for linearly-constrained composite optimization.

6. Numerical Experiments. This section presents experiments that demonstrate the numerical efficiency of PF.APD. Comments about the results are given in Subsection 6.4.

We first describe the benchmark algorithms, the implementation of APD, and the computing environment. The benchmark algorithms are instances of PGD, AIPP, ANCF, and UPF described in Section 1 and Table 1.1. Specifically, AIPP uses $\sigma = 1/4$, ANCF uses $\theta = 1.25$, and UPF uses $\gamma_1 = \gamma_2 = 0.4$, $\gamma_3 = 1$, $\beta_0 = 1$, and $\hat{\lambda}_0 = 1$. Moreover, UPF uses $\hat{\lambda}_k$ for the initial estimate of $\hat{\lambda}_{k+1}$ for $k \geq 1$ and AIPP stops its call of ACG when the condition $\|u_j\|^2 + 2\eta_j \leq \sigma\|x_0 - x_j + u_j\|^2$ holds (inside of ACG) instead of prescribing a fixed number of ACG iterations. The implementations for ANCF and UPF were generously provided by the respective authors of [25] and [13], while the author implemented AIPP and PGD.[8] Note that we did not consider the VAR-FISTA method in [43] because: (i) its steps were similar to ANCF and (ii) we already had a readily available and optimized code for the ANCF method.

The implementation of PF.APD, abbreviated as APD, is as in Algorithm 3.4 with $\alpha = \beta = 2$, $\hat{m} = m_k$ for every $k \geq 1$, and the following additional updates at the beginning of every call to Algorithm 3.2 and the $(k+1)^{\text{th}}$ iteration of Algorithm 3.4, respectively:

$$L_0 \leftarrow \frac{L_0}{1+\beta/2}, \quad m_{k+1} \leftarrow \max\left\{m_0, \frac{m_{k+1}}{1+\alpha/2}\right\}. \tag{6.1}$$

This is done to allow a possible decrease in both of the curvature estimates. While we do not show convergence of this modified PF.APD, we believe that convergence

[8]See `https://github.com/wwkong/nc_opt/tree/master/tests/papers/apd` for the source code of the experiments.

can be established using similar techniques as in [35]. It is worth mentioning that the modification in (6.1) substantially improves upon the numerical performance of PF.APD compared to the version given in Algorithm 3.4.

All experiments were run in MATLAB 2023a under a 64-bit Windows 11 machine with an Intel Core i7-10700K processor and 16 GB of RAM. All algorithms except AIPP use an initial curvature estimate of $(m_0, M_0) = (1, 1)$, and each algorithm stops when it finds a pair $(\bar{z}, \bar{v})$ solving Problem $\mathcal{CO}$ for some $\varepsilon > 0$. A time limit of 1200 (resp. 2400) seconds was prescribed for the problems in Subsection 6.1 and 6.3 (resp. Subsection 6.2). We also set an (innermost) iteration limit of 500000 (resp. 10000) for Subsection 6.2 (resp. Subsection 6.3).

6.1. Quadratic Semidefinite Programming. The problem of interest is the 400-variable nonconvex quadratic semidefinite programming (QSDP) problem

$$\begin{aligned} &\min_{Z \in \mathbb{R}^{35\times 35}} \; -\frac{\eta_1}{2}\|D\mathcal{B}(Z)\|_2^2 + \frac{\eta_2}{2}\|\mathcal{A}(Z) - b\|_2^2, \\ &\quad \text{s.t. } \operatorname{tr}(Z) = 1, \quad Z \in \mathcal{S}_+^{35}, \end{aligned} \tag{6.2}$$

where $\mathcal{S}_+^n$ is the n-dimensional positive semidefinite cone, $\operatorname{tr}(Z)$ is the trace of a matrix, $b \in \mathbb{R}^{10}$, $D \in \mathbb{R}^{10\times 10}$ is a diagonal matrix with nonzero entries randomly generated from $\{1, ..., 1000\}$, $(\eta_1, \eta_2) \in \mathbb{R}^2_{++}$ are chosen to yield a particular curvature pair, and $\mathcal{A}, \mathcal{B} : \mathcal{S}_+^{20} \mapsto \mathbb{R}^{10}$ are linear operators defined by

$$[\mathcal{A}(Z)]_j = A_j \bullet Z, \quad [\mathcal{B}(Z)]_j = B_j \bullet Z$$

for matrices $\{A_j\}_{j=1}^{10}, \{B_j\}_{j=1}^{10} \subseteq \mathbb{R}^{20\times 20}$. Moreover, the entries in these matrices and b were sampled from the uniform distribution on $[0, 1]$.

To build the decomposition in (1.1), we set f equal to the objective function of (6.2), h equal to the indicator function of the constraint set of (6.2). The starting point was set to $z_0 = I_{20}/20$, where I_{20} is an identity matrix, and the tolerance was set to $\varepsilon = 10^{-6}(1 + \|\nabla f(z_0)\|_2)$.

	# of Function Evaluations				# of Gradient Evaluations				Runtime (seconds)			
m, M	UPF	ANCF	AIPP	APD	UPF	ANCF	AIPP	APD	UPF	ANCF	AIPP	APD
$10^2, 10^4$	6.5E4	2.1E4	7.1E4	**1.1E3**	1.3E4	1.6E4	6.7E4	**2.1E3**	9.2E1	2.7E1	1.1E2	**3.2E0**
$10^2, 10^5$	1.9E5	4.4E4	4.1E5	**3.3E3**	3.8E4	3.3E4	3.9E5	**6.7E3**	2.6E2	5.8E1	6.5E2	**9.9E0**
$10^2, 10^6$	3.0E5	5.9E4	7.6E5	**7.1E3**	6.1E4	4.4E4	7.0E5	**1.4E4**	4.3E2	7.9E1	1.2E3	**2.1E1**
$10^3, 10^7$	3.0E5	5.9E4	7.6E5	**1.0E4**	6.1E4	4.4E4	6.9E5	**2.0E4**	4.3E2	8.1E1	1.2E3	**3.0E1**
$10^2, 10^7$	3.3E5	6.6E4	2.6E5	**1.2E4**	6.5E4	5.0E4	1.3E5	**2.4E4**	4.5E2	8.6E1	2.5E2	**3.4E1**
$10^1, 10^7$	5.8E5	1.4E5	8.8E4	**2.0E4**	1.2E5	1.1E5	4.4E4	**4.1E4**	7.9E2	1.9E2	8.3E1	**5.8E1**

TABLE 6.1
Unique function evaluations, unique gradient evaluations, and runtimes in the QSDP experiments for different curvature pairs (m, M). The bolded numbers indicate the best algorithm in terms of the number of evaluations (less is better) and runtime (less is better). Entries marked with "-" are those that did not terminate within the prescribed time limit.

Table 6.1 reports the number of unique function evaluations, unique gradient evaluations, and runtime (in seconds) for different curvature pairs (m, M), and Figure 6.1 plots the minimum norm of the normalized stationarity residual $\|\bar{v}\|$ over iteration count for each algorithm and curvature pairs $(m, M) = (10^2, 10^4)$, $(10^2, 10^5)$, and $(10^2, 10^6)$.

6.2. Sparse Vector Recovery. The problem of interest is the penalized sparse vector recovery (SVR) problem [45]

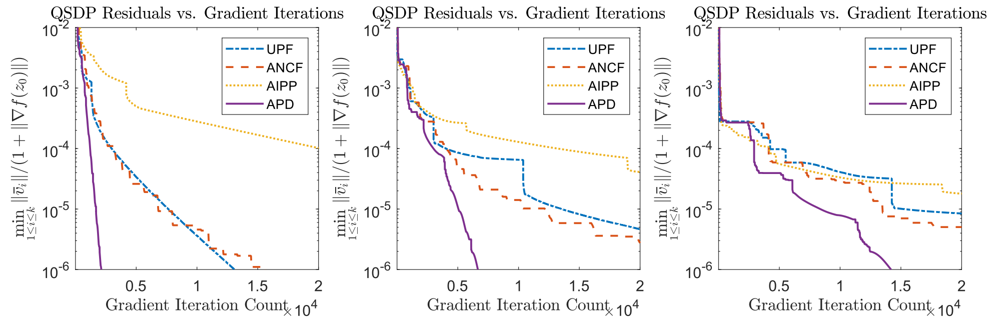

FIGURE 6.1. *Plots of the minimum norm of the normalized stationarity residual* $\|\bar{v}\|$ *over iteration count in the QSDP experiments. The curvature pairs for the plots are* $(10^2, 10^4)$, $(10^2, 10^5)$, *and* $(10^2, 10^6)$ *from left-to-right.*

$$\min_{z\in\mathbb{R}^n} \frac{1}{2}\|Az-b\|_2^2 + \frac{\tau}{2}\|z\|_2^2 + \mathrm{LPL}_{\gamma,\delta}(\|z\|_2) \tag{6.3}$$

where $\tau = 10^{-2}$, $A \in \mathbb{R}^{\ell\times p}$ with $\ell \geq p$, $b = A\tilde{u}$ where u is a random vector whose entries are sampled uniformly from [0,1], for $(\gamma,\delta) = (10, 10^{-1})$, the function $\mathrm{LPL}_{\gamma,\delta}(z) = \gamma[1-\exp(-z/\delta)]$ is the concave Laplace penalty function [44] at z. The goal of this problem is to find a sparse vector $\hat{z}$ such that $A\hat{z}$ is close to b.

Each matrix A is built from a recommender dataset where each entry corresponds to a user-item rating. Specifically, the datasets were taken from the well-known Jester, MovieLens 100K, and FilmTrust datasets and the musical instruments and patio, lawn, and garden products Amazon Review datasets published by the University of California San Diego. The dimensions (ℓ, p) of each matrix generated by the previous datasets were $(24938, 100)$, $(9724, 610)$, $(2071, 1508)$, $(1429, 900)$, $(1686, 962)$, respectively.

To put (6.3) into the form of (1.1), we use the decomposition given in [46] where h is a multiple of the 1-norm and f is the function in (6.3) minus h. The starting point z_0 was set to be a vector whose entries are all equal to p, and the tolerance was set to $\varepsilon = 10^{-10}(1 + \|\nabla f(z_0)\|_2)$. Following the analysis in [46], AIPP uses the curvature pair $(m, M) = (2\gamma/\delta^2, \tau + \sigma^2_{\max}(A))$, where $\sigma_{\max}(A)$ is the largest singular value of A.

	# of Function Evaluations				# of Gradient Evaluations				Runtime (seconds)			
ℓ, p	UPF	ANCF	AIPP	APD	UPF	ANCF	AIPP	APD	UPF	ANCF	AIPP	APD
1429, 900	6.9E3	3.5E3	1.5E4	**3.7E2**	8.0E2	2.6E3	1.1E4	**7.3E2**	2.7E0	1.2E0	1.1E1	**3.4E-1**
1686, 962	2.9E4	1.1E4	7.7E4	**2.6E3**	4.9E3	8.2E3	5.8E4	**3.8E3**	1.3E1	4.1E0	6.0E1	**2.4E0**
9724, 610	3.9E4	4.3E4	6.2E4	**3.2E3**	6.3E3	3.2E4	3.3E4	**6.2E3**	3.6E1	3.5E1	8.4E1	**6.0E0**
24938, 100	5.7E5	2.4E5	9.8E5	**2.5E4**	1.1E5	1.8E5	5.0E5	**4.8E4**	1.7E2	5.0E1	4.3E2	**1.6E1**
2071, 1508	-	2.9E5	-	**2.8E4**	-	2.2E5	-	**5.5E4**	-	1.3E3	-	**2.6E2**

TABLE 6.2

Unique function evaluations, unique gradient evaluations, and runtimes in the SVR experiments for different datasets and their dimensions (ℓ, p). *The bolded numbers indicate the best algorithm in terms of the number of evaluations (less is better) and runtime (less is better). Entries marked with "-" are those that did not terminate within the prescribed time or iteration limit.*

Table 6.2 reports the unique function evaluations, unique gradient evaluations, and runtime (in seconds) for the different datasets mentioned above, and Figure 6.2

plots the minimum norm of the normalized stationarity residual $\|\bar{v}\|$ over the gradient count for each algorithm and the first, second, and fourth row of Table 6.2.

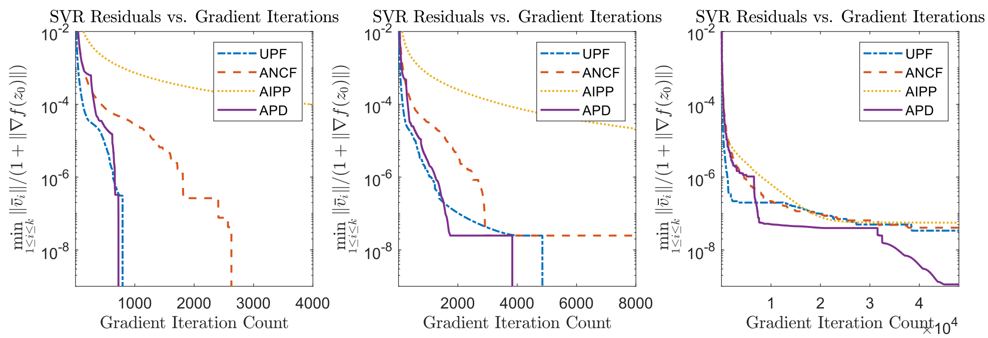

FIGURE 6.2. *Plots of the minimum norm of the normalized stationarity residual $\|\bar{v}\|$ over iteration count in the SVR experiments. The dimensions and upper curvature (ℓ, p) for the plots are* $(1429, 900)$*,* $(1686, 962)$*, and* $(24938, 100)$ *from left-to-right.*

6.3. Low-Rank Matrix Completion. The problem of interest is the penalized nonconvex low-rank matrix completion (LRMC) problem [45, 46]

$$\min_{Z \in \mathbb{R}^{\ell \times p}} \frac{1}{2}\|\Pi_\Omega(Z) - \Pi_\Omega(X)\|_F^2 + \frac{\tau}{2}\|Z\|_F^2 + (\mathrm{MCP}_{\gamma,\delta} \circ \sigma)(Z), \tag{6.4}$$

where $\tau = 10^{-7}$, $X \in \mathbb{R}^{\ell \times p}$ is a reference image, $\sigma : \mathbb{R}^{\ell \times p} \mapsto \mathbb{R}^{\min\{\ell,p\}}$ maps a matrix to its vector of singular values, for $(\gamma, \delta) = (450, 10^{-4})$ the function $\mathrm{MCP}_{\gamma,\delta}(z)$ is the minimax concave penalty (MCP) function [47] at z (which takes value $\gamma z - z^2/(2\delta)$ if $z \le \gamma\delta$ and $\gamma^2\delta/2$ otherwise), and, for a given corrupted image Ω, the function $\Pi_\Omega : \mathbb{R}^{\ell \times p} \mapsto \mathbb{R}^{\ell \times p}$ is the projection operator that zeros out entries of its input where the corresponding entry in Ω is zero. The goal of this problem is to fill in the zero entries of a corrupted image Ω of X so that the resulting image $\hat{Z}$ is close to X.

To put (6.4) into the form of (1.1), we use the decomposition given in [46] where h is a multiple of the nuclear norm and f is the function in (6.4) minus h. Experiments were run on different reference images X given in the first row of Figure 6.3 and Ω was set to be a corrupted version of X where we add Gaussian noise with a 100 dB signal-to-noise ratio and remove 30% of the resulting pixels. For illustration, two corrupted images can be found in the first columns of the last two rows in Figure 6.3. The starting point Z_0 was set to be a matrix whose entries were equal to the average of the grayscale value of Ω, and the tolerance was set to $\varepsilon = 10^{-10}(1 + \|\nabla f(Z_0)\|_F)$. Following the analysis in [46], AIPP uses the curvature pair $(m, M) = (2/\delta, 1 + \tau)$.

Table 6.3 presents the relative error[9] of the final candidate image and runtime (in seconds) for the different reference images, and the last two rows in Figure 6.3 show the candidate images generated by each method for two of the reference images.

6.4. Comments about the numerical results. In Subsection 6.1, APD substantially outperformed[10] its competitors and its non-adaptive variant AIPP under

[9] For a candidate image $\hat{Z}$, this quantity is defined as $\|\hat{Z} - X\|_F$ divided by $\max_{Z \in \Xi} \|Z - X\|_F$ where Ξ is the set of all grayscale images. Its value can range from 0.0 (full recovery) to 1.0.

[10] 5-20x (resp. 2-7x) fewer function (resp. gradient) evaluations for ANCF and 27-60x (resp. 2-6x) fewer for UPF.

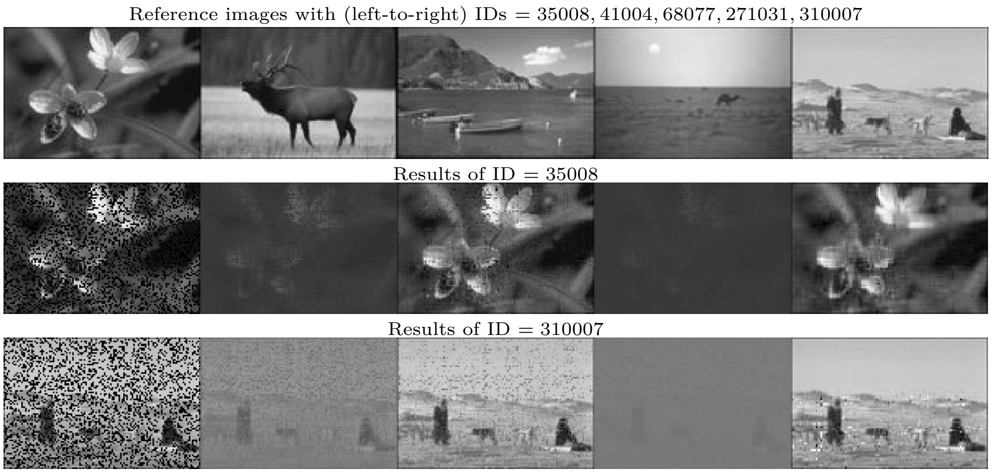

FIGURE 6.3. *The first row presents the downscaled (80×120) reference images X taken from the Berkeley Segmentation Dataset, along with their image IDs (in order). The second and third rows present the results of the LRMC experiments for two of the images. Specifically, each of these rows presents (from left to right) the corrupted image Ω and the images generated by UPF, ANCF, AIPP, and APD, respectively.*

	Relative Error				Runtime (seconds)			
image id	UPF	ANCF	AIPP	APD	UPF	ANCF	AIPP	APD
35008	0.220	0.059	0.241	**0.034**	104.7	174.5	89.2	**44.4**
41004	0.259	0.103	0.312	**0.072**	114.6	175.9	90.6	**45.9**
68077	0.238	0.075	0.276	**0.046**	107.6	175.6	89.2	**43.0**
271031	0.272	0.146	0.363	**0.079**	117.5	176.8	95.7	**48.0**
310007	0.265	0.079	0.324	**0.048**	116.0	186.4	92.7	**44.9**

TABLE 6.3
Relative errors and runtimes in the LRMC experiments for different reference images in the LRMC experiments. The bolded numbers indicate the best algorithm in terms of the relative error (less is better) and runtime in seconds (less is better).

the given numerical tolerance ε. However, Figure 6.1 showed that ANCF was more comparable to PF.APD when the curvature ratio M/m was large or a larger (more lenient) tolerance was given. In Subsection 6.2, APD consistently outperformed its competitors on all metrics. For the number of gradient evaluations, UPF performed similarly to APD but was among the worst adaptive methods for function evaluations. In Subsection 6.3, APD generated higher-quality candidate images compared to its competitors under a fixed iteration budget. Specifically, it was shown in Figure 6.3 that PF.APD generated images with fewer artifacts, more consistent lighting, and in a more timely manner.

7. Concluding Remarks. This paper establishes iteration complexity bounds for PF.APD that are only optimal, up to logarithmic terms, in terms of $(M, \Delta_0, \varepsilon)$ when f is convex and in terms of $(m, M, \Delta_0, \varepsilon)$ when f is weakly-convex. Consequently, it remains to be seen whether an optimal complexity bound in terms of d_0 exists for a parameter-free and convexity-unaware method.

To alleviate the issues regarding the d_0-suboptimal complexity of APD — specifically, when f is convex and d_0 — one could consider running running $S+1$ instances of PF.APD (either in lockstep or in parallel) with different initial estimates $m_0 = 1, \varepsilon, \varepsilon/2, \ldots, \varepsilon/2^{S-1}$; in particular, the whole scheme stops when one of these instances stops successfully. The number of resolvent evaluations of this approach is at most $S + 1$ times the minimum of the bound in (3.19) over the different values of m_0. Consequently, following the remarks at the end of Section 3, if $d_0 \le 2^{S-1}$ then one of the $S + 1$ instances obtains the lower bound in Table 1.1 for the convex case; otherwise, the bound for APD in Table 1.1 is obtained. Moreover, if S is chosen small compared to the other terms in (3.19) and $d_0 \le 2^{S-1}$, then the cost is on the same order of magnitude as the $(M, \Delta_0, d_0, \varepsilon)$-complexity optimal method described at the end of Section 3 (which requires knowledge of d_0).

In addition to the applications in Section 5, it would be interesting to see if PF.APD could be leveraged to develop a parameter-free proximal augmented Lagrangian method, following schemes similar to ones as in [20, 27].

REFERENCES

[1] M. Ahookhosh and A. Neumaier, *Solving structured nonsmooth convex optimization with complexity $\mathcal{O}(\varepsilon^{-1/2})$*, TOP, 26 (2018), pp. 110–145.

[2] M. M. Alves, R. D. C. Monteiro, and B. F. Svaiter, *Regularized HPE-type methods for solving monotone inclusions with improved pointwise iteration-complexity bounds*, SIAM J. Optim., 26 (2016), pp. 2730–2743.

[3] H. H. Bauschke, P. L. Combettes, et al., *Convex analysis and monotone operator theory in Hilbert spaces*, vol. 408, Springer, 2011.

[4] A. Beck, *First-order methods in optimization*, SIAM, 2017.

[5] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sci., 2 (2009), pp. 183–202.

[6] Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford, *Accelerated methods for nonconvex optimization*, SIAM J. Optim., 28 (2018), pp. 1751–1772.

[7] Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford, *Lower bounds for finding stationary points II: first-order methods*, Math. Program., 185 (2021), pp. 315–355.

[8] D. Davis and D. Drusvyatskiy, *Stochastic model-based minimization of weakly convex functions*, SIAM J. Optim., 29 (2019), pp. 207–239.

[9] D. Drusvyatskiy, *The proximal point method revisited*, arXiv preprint arXiv:1712.06038, (2017).

[10] D. Drusvyatskiy and C. Paquette, *Efficiency of minimizing compositions of convex functions and smooth maps*, Math. Program., 178 (2019), pp. 503–558.

[11] M. I. Florea and S. A. Vorobyov, *An accelerated composite gradient method for large-scale composite objective problems*, IEEE Trans. Signal Process., 67 (2018), pp. 444–459.

[12] S. Ghadimi and G. Lan, *Accelerated gradient methods for nonconvex nonlinear and stochastic programming*, Math. Program., 156 (2016), pp. 59–99.

[13] S. Ghadimi, G. Lan, and H. Zhang, *Generalized uniformly optimal methods for nonlinear programming*, J. Sci. Comput., 79 (2019), pp. 1854–1881.

[14] S. Guminov, P. Dvurechensky, N. Tupitsa, and A. Gasnikov, *On a combination of alternating minimization and Nesterov's momentum*, in Int. Conf. Mach. Learn., PMLR, 2021, pp. 3886–3898.

[15] S. Guminov, Y. Nesterov, P. Dvurechensky, and A. Gasnikov, *Accelerated primal-dual gradient descent with linesearch for convex, nonconvex, and nonsmooth optimization problems*, in Dokl. Math., vol. 99, Springer, 2019, pp. 125–128.

[16] W. Hare and C. Sagastizábal, *A redistributed proximal bundle method for nonconvex optimization*, SIAM J. Optim., 20 (2010), pp. 2442–2473.

[17] W. Kong, *Accelerated inexact first-order methods for solving nonconvex composite optimization problems*, arXiv preprint arXiv:2104.09685, (2021).

[18] W. Kong, J. G. Melo, and R. D. C. Monteiro, *Complexity of a quadratic penalty accelerated inexact proximal point method for solving linearly constrained nonconvex composite programs*, SIAM J. Optim., 29 (2019), pp. 2566–2593.

[19] W. Kong, J. G. Melo, and R. D. C. Monteiro, *An efficient adaptive accelerated inexact proximal point method for solving linearly constrained nonconvex composite problems*, Comput. Math. Appl. , 76 (2020), pp. 305–346.

[20] W. Kong, J. G. Melo, and R. D. C. Monteiro, *Iteration-complexity of a proximal augmented Lagrangian method for solving nonconvex composite optimization problems with nonlinear convex constraints*, arXiv preprint arXiv:2008.07080, (2020).

[21] W. Kong, J. G. Melo, and R. D. C. Monteiro, *FISTA and extensions-review and new insights*, arXiv preprint arXiv:2107.01267, (2021).

[22] W. Kong and R. D. C. Monteiro, *An accelerated inexact proximal point method for solving nonconvex-concave min-max problems*, SIAM J. Optim., 31 (2021), pp. 2558–2585.

[23] H. Li and Z. Lin, *Accelerated proximal gradient methods for nonconvex programming*, Adv. Neural Inf. Process. Syst., 28 (2015).

[24] J. Liang and R. D. C. Monteiro, *A doubly accelerated inexact proximal point method for nonconvex composite optimization problems*, arXiv preprint arXiv:1811.11378, (2018).

[25] J. Liang, R. D. C. Monteiro, and C.-K. Sim, *A FISTA-type accelerated gradient algorithm for solving smooth nonconvex composite optimization problems*, Comput. Math. Appl. , 79 (2021), pp. 649–679.

[26] M. Marques Alves, R. D. C. Monteiro, and B. F. Svaiter, *Iteration-complexity of a Rockafellar's proximal method of multipliers for convex programming based on second-order approximations*, Optimization, 68 (2019), pp. 1521–1550.

[27] J. G. Melo, R. D. C. Monteiro, and W. Kong, *Iteration-complexity of an inner accelerated inexact proximal augmented Lagrangian method based on the classical lagrangian function and a full Lagrange multiplier update*, arXiv preprint arXiv:2008.00562, (2020).

[28] R. D. C. Monteiro, C. Ortiz, and B. F. Svaiter, *An adaptive accelerated first-order method for convex optimization*, Comput. Math. Appl. , 64 (2016), pp. 31–73.

[29] R. D. C. Monteiro, M. R. Sicre, and B. F. Svaiter, *A hybrid proximal extragradient self-concordant primal barrier method for monotone variational inequalities*, SIAM J. Optim., 25 (2015), pp. 1965–1996.

[30] R. D. C. Monteiro and B. F. Svaiter, *Convergence rate of inexact proximal point methods with relative error criteria for convex optimization*, Optimization Online preprint, (2010).

[31] R. D. C. Monteiro and B. F. Svaiter, *On the complexity of the hybrid proximal extragradient method for the iterates and the ergodic mean*, SIAM J. Optim., 20 (2010), pp. 2755–2787.

[32] Y. Nesterov, *A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$*, Dokl. Akad. Nauk, 269 (1983), pp. 543–547.

[33] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*, vol. 87, Springer Science & Business Media, 2003.

[34] Y. Nesterov, *How to make the gradients small*, Optim. Math. Optim. Soc. Newsl., (2012), pp. 10–11.

[35] Y. Nesterov, *Gradient methods for minimizing composite functions*, Math. Program., 140 (2013), pp. 125–161.

[36] Y. Nesterov, *Universal gradient methods for convex optimization problems*, Math. Program., 152 (2015), pp. 381–404.

[37] Y. Nesterov, *Lectures on convex optimization*, vol. 137, Springer, 2 ed., 2018.

[38] Y. Nesterov, A. Gasnikov, S. Guminov, and P. Dvurechensky, *Primal–dual accelerated gradient methods with small-dimensional relaxation oracle*, Optim. Methods Softw, (2020), pp. 1–38.

[39] A. Neumaier, *Osga: a fast subgradient algorithm with optimal complexity*, Math. Program., 158 (2016), pp. 1–21.

[40] C. Paquette, H. Lin, D. Drusvyatskiy, J. Mairal, and Z. Harchaoui, *Catalyst acceleration for gradient-based non-convex optimization*, arXiv preprint arXiv:1703.10993, (2017).

[41] N. Parikh, S. Boyd, et al., *Proximal algorithms*, Found. Trends Optim., 1 (2014), pp. 127–239.

[42] R. T. Rockafellar, *Augmented Lagrangians and applications of the proximal point algorithm in convex programming*, Math. Oper. Res., 1 (1976), pp. 97–116.

[43] C.-K. Sim, *A FISTA-type first order algorithm on composite optimization problems that is adaptable to the convex situation*, arXiv preprint arXiv:2008.09911, (2020).

[44] J. Trzasko and A. Manduca, *Highly undersampled magnetic resonance image reconstruction via homotopic ℓ_0-minimization*, IEEE Trans. Med. Imaging, 28 (2008), pp. 106–121.

[45] F. Wen, L. Chu, P. Liu, and R. C. Qiu, *A survey on nonconvex regularization-based sparse and low-rank recovery in signal processing, statistics, and machine learning*, IEEE Access, 6 (2018), pp. 69883–69906.

[46] Q. Yao and J. Kwok, *Efficient learning with a family of nonconvex regularizers by redistributing nonconvexity*, in Int. Conf. Mach. Learn., PMLR, 2016, pp. 2645–2654.

[47] C.-H. Zhang, *Nearly unbiased variable selection under minimax concave penalty*, Ann. Statist., 38 (2010), pp. 894–942.

[48] D. Zhou and Q. Gu, *Lower bounds for smooth nonconvex finite-sum optimization*, in Int. Conf. Mach. Learn., PMLR, 2019, pp. 7574–7583.